Foundations and Trends® in Systems and Control
Vol. XX, No. XX (2015) 1–79
© 2015
DOI: 10.1561/XXXXXXXXXX

Operator splitting methods in control

XXX
École Polytechnique Fédérale de Lausanne (EPFL)
XXX

XXX
École Polytechnique Fédérale de Lausanne (EPFL)
XXX

Contents

Abstract

The significant progress that has been made in recent years both in hardware implementations and in numerical computing has rendered real-time optimization-based control a viable option when it comes to advanced industrial applications. More recently, the need for control of a process in the presence of a limited amout of hardware resources has triggered research in the direction of embedded optimization-based control. At the same time, and standing at the other side of the spectrum, the field of big data has emerged, seeking for solutions to problems that classical optimization algorithms are incapable to provide. This triggered some interest to revisit the family of first order methods commonly known as *decomposition schemes* or *operator splitting methods*. Although it is established that splitting methods are quite beneficial when applied to large-scale problems, their potential in solving small to medium scale embedded optimization problems has not been studied so extensively. Our purpose is to study the behavior of such algorithms as solvers of control-related problems of that scale. Our effort focuses on identifying special characteristics of these problems and how they can be exploited by some popular splitting methods.

. *Operator splitting methods in control.* Foundations and Trends® in Systems and Control, vol. XX, no. XX, pp. 1–79, 2015.

DOI: 10.1561/XXXXXXXXXX.

1 Introduction

The significant progress that has been made in recent years both in hardware implementations and in numerical computing has rendered real-time optimization-based control a viable option when it comes to advanced industrial applications. More recently, the need for control of a process in the presence of a limited amout of hardware resources has triggered research in the direction of embedded optimization-based control. Many efficient high-speed solvers have been developed for both linear and nonlinear control, based on either *first order methods* (FiOrdOs [87], QPgen [38],[39], DuQuad [58]), *interior point (IP) methods* (FORCES [27], CVXGEN [56]) and *active set methods* (QPOASES [33]).

In this work we focus on systems with linear dynamics, giving rise to convex control problems. The purpose of the survey is to explore a family of first order methods known as *decomposition schemes* or *operator splitting methods*. The abstract form of the problem at hand is the minimization of the sum of two convex functions subject to linear equality constraints, and can be written as

$$\text{minimize} \quad f(z) + g(Lz) \ , \tag{1.1}$$

with variables $z \in \mathbb{R}^n$, where $f \in \Gamma_0(\mathbb{R}^n)$ and $g \in \Gamma_0(\mathbb{R}^p)$ and $A : \mathbb{R}^n \to \mathbb{R}^p$ is a linear map. A splitting method can be applied to the above problem after rewriting it as

$$\begin{array}{ll} \text{minimize} & f(z) + g(y) \\ \text{subject to} & Lz = y \ , \end{array} \tag{1.2}$$

by alternatively (or simultaneously) minimizing over f and g. A dual variable update for the equality constraint ensures that the solutions of problems (1.2) and (1.1) are identical. Inequality constraints that might appear are already embedded in one of the two functions in the form of indicator functions, *i.e.*, a membership function for a set $\mathcal{C}$

$$\delta_{\mathcal{C}}(z) = \begin{cases} 0 & z \in \mathcal{C} \\ \infty & \text{otherwise,} \end{cases} \tag{1.3}$$

which is the reason why both f and g are considered to be *extended-value functions* (see [12, Section 3.1.2]). Formulations similar to the above have been studied extensively and we can look for their roots in the method of multipliers [53], [76], the Arrow-Hurwicz method [54], Douglas-Rachford splitting [28] and ADMM [40], [35]. Decomposition of the original problem into simpler ones is beneficial when distributed computation tools are available. This potential is already suggested in the classical references [10] and [29]. It was not until recently, though, that decomposition algorithms were indeed applied in modern engineering problems (signal and image processing, big data analysis, machine learning, [13] and [19]), in cases where off-the-shelf interior point solvers simply fail due to the large dimensions involved. The thesis [31] provides a comprehensive description of the connection of several splitting algorithms under a common framework. Finally, the book [3] provides a mathematically rigorous introduction to operator splitting methods in general Hilbert spaces.

The plethora of different approaches for solving problem (1.2) is partly a consequence of the problem-dependent behavior of first-order methods. This behavior has both its pros and cons; on one hand, sensitivity to the problem's structure and data requires preprocessing and tuning of several parameters, a procedure that can be cumbersome.

However, it is exactly this procedure that gives the flexibility to customize the solver to the problem at hand, and, in many cases, outperform by several orders of magnitude general purpose solvers. Consequently, there are numerous approaches, each of which can be less or more pertinent for the specific problem. Mentioning some of the most important categorizations, we can solve either the *primal* problem, the *dual* problem, or a *primal-dual* formulation. Regarding primal approaches, the most popular one is the primal decomposition method [10], [14], where the original problem is decomposed into a master problem and two subproblems. Primal decomposition works well when the complicating variables for the two subproblems are few. Dualization plays a crucial role in more complicated problems. It can be performed by means of *Lagrangian relaxations* (dual decomposition [24], [32], [83], [9]), *augmented Lagrangian relaxations* [8], [81], [80], *alternating minimization (Gauss-Seidel) augmented Lagrangian schemes* (ADMM), mixture of Lagrangian with augmented Lagrangian schemes (AMA [85]), *linearized augmented Lagrangians* or *approximate minimization* schemes (L-PMM [17], PADMM [1]) and, finally, *mixtures of alternating minimization with partial linearization* (PDHG [55], [30], Chambolle-Pock algorithm [16], [23] and several similar primal-dual schemes [20], [88], [11]).

Although it is well-established that splitting methods are quite beneficial when applied to large-scale problems, their potential in solving small to medium scale embedded optimization problems has not been studied in so extensively. It was not until very recently that the first works attempting to apply decomposition methods in control problems started making their appearance [69], [36], [38], [39], [73]. Our purpose is to study the behavior of such algorithms as solvers of control-related convex problems of that scale. Our effort focuses on identifying special characteristics of these problems and how they can be exploited from some popular splitting methods. Some of the questions that we attempt to answer are:

1. It is very common in practice that optimal control problems come with a quadratic objective, since in this way stability can be proven for regulation or tracking purposes. What is the best way

to exploit this smooth term, along with the special structure of the dynamics equation?

2. Given that a control problem has to be solved repeatedly (*e.g.*, MPC), how does warm-starting of the solution affect the speed?
3. Given the structure of the problem at hand, which algorithms will converge more quickly?
4. Are there ways to precondition the problem in order to reduce the solve time?

In what follows we present three well-understood splitting algorithms, the *alternating direction method of multipliers (ADMM)*, the *alternating minimization algorithm (AMA)* and a *primal-dual algorithms (PDA)*, the most popular representative of several primal-dual schemes that have been recently developed. These three methods come from different sides of the spectrum described above, but also hold very strong similarities. Our choice is motivated from the fact that the methods are analyzed and extended from several communities, and hence their properties are well-understood.

The paper is organized as follows: In Chapter 2 we formulate the problem we want to solve and look at it from three different perspectives, resulting to the three algorithms we use. Subsequently we introduce the algorithms under a unified scheme and report their properties. In the next two chapters we get past the basic variants of the methods presented before, we introduce several enhanced versions and we focus on their applicability to solving optimization problems. More specifically, in Chapter 3 we review how one can exploit the structure of the problem to accelerate the theoretical convergence rates. In Chapter 4 we extend the discussion on acceleration to more practical schemes, *i.e.*, stepsize selection and preconditioning. We provide a comprehensive literature review of existing methods and we present generic preconditioned versions of the three algorithms. In Chapter 5 we discuss the computational aspects; we identify the bottlenecks in each method and propose ways to speed up the computation. In Chapter 6 we summarize the observations that we have made and attempt to construct a guideline about how to choose a splitting scheme given a problem. Finally, the algorithms are illustrated with three examples in Chapter 6.

1.1 Notation and Definitions

Let $\mathcal{Z} = (\mathbb{R}^n, \langle \cdot, \cdot \rangle)$ be a Euclidean space equipped with the inner product $\langle z, x \rangle = z^T x$ and the corresponding norm $\|z\| = \sqrt{\langle z, z \rangle}$. Symmetric n-dimensional matrices are denoted with $\mathbb{S}^n$, while positive (semi)definite matrices are denoted with $(\mathbb{S}^n_+)\mathbb{S}^n_{++}$. We also consider the scaled norm $\|z\|_P = \sqrt{\langle z, Pz \rangle}$, with $P \in \mathbb{S}_+$. The matrix norm of the linear operator $M \in \mathbb{R}^{m \times n}$ is defined as $\|M\| = \sup\limits_{z \neq 0} \frac{\|Mz\|_2}{\|z\|_2}$. The minimum and maximum eigenvalue of a matrix $Q \in \mathbb{R}^{n \times n}$ are denoted by $\lambda_{\min}(Q)$ and $\lambda_{\max}(Q)$, respectively.
The domain of the real-valued function f is denfined as $\mathbf{dom}\, f = \{z \in \mathcal{Z} : f(z) < +\infty\}$ and f is proper if $\mathbf{dom}\, f \neq \emptyset$. The function f is closed if its epigraph $\mathbf{epi}\, f = \{(z, t) \in \mathbb{R}^n \times \mathbb{R} : f(z) \leq t\}$ is a closed nonempty convex set. The range of extended real-valued functions is denoted with $\mathbb{R} \cup \{+\infty\} = \overline{\mathbb{R}}$. We denote the conjugate of a convex function with $f^\star$, while the optimizer is denoted by the asterisk, *i.e.*, $f(z^*) \leq f(z)\ \forall z \in \mathcal{Z}$. Finally, for succinctness in the notation, we denote the class of all proper, closed, convex functions from $\mathcal{Z}$ to $\overline{\mathbb{R}}$ with $\Gamma_0(\mathcal{Z})$.

2

The algorithms

We narrow the general formulation to our problems of interest which can, without loss of generality, be written as

$$\begin{array}{ll} \text{minimize} & (1/2)z^TQz + c^Tz + \sum\limits_{i=1}^{M} g_i(L_iz + l_i) \\ \text{subject to} & Az = b \ , \end{array} \tag{P}$$

with variable $z \in \mathbb{R}^n$ and data $Q \in \mathbb{S}_+^n$, $L_i \in \mathbb{R}^{p_i \times n}$, $l_i \in \mathbb{R}^{p_i}$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The following assumption holds:

Assumption 1. The functions $g_i : \mathbb{R}^{p_i} \to \overline{\mathbb{R}}$ are closed, proper, convex functions, *i.e.*, $g_i \in \Gamma_0(\mathbb{R}^{p_i})$.

Formulation (P) is quite general and can describe any convex optimization problem. The choice of the quadratic part $(1/2)z^TQz + c^Tz$ and the equality constraints $Az = b$ being represented in an explicit way is motivated by the standard form of control problems. The constraints are usually expressed through indicator functions g_i.

It is important to mention that the original formulation (1.2) involves two functions in the objective, while in (P) we consider *two* *groups of functions.* The first group contains two functions expressed as

$$f(z) := h(z) + \delta_{\mathcal{D}}(z) \ , \tag{2.1}$$

where $h(z) = (1/2)z^TQz + c^Tz$, $h : \mathbb{R}^n \to \mathbb{R}$ and $f : \mathbb{R}^n \to \overline{\mathbb{R}}$. Note that we use the indicator function

$$\delta_{\mathcal{D}}(z) = \begin{cases} 0 & Az = b \\ \infty & \text{otherwise.} \end{cases}$$

to restrict h to the subspace spanned by the dynamics equation. The second group constitutes of M functions $g_i(y_i)$. By introducing slack variables $y_i = L_iz + l_i, \ i = 1, \ldots, M$, and subsequently concatenating the vectors and matrices associated with the affine terms in the $g_i(\cdot)$ functions as $L = (L_1, \ldots, L_M)$ and $l = (l_1, \ldots, l_M)$, we can recast (P) as

$$\begin{array}{ll} \text{minimize} & f(z) + g(y) \\ \text{subject to} & Lz - y = -l \ , \end{array}$$

where $g(y) = \sum_{i=1}^{M} g_i(y_i)$, $g : \mathbb{R}^{p_1} \times \cdots \times \mathbb{R}^{p_M} \to \overline{\mathbb{R}}$. Thus we end up with the original formulation (1.2). Note that it is possible to proceed with such a scheme because the variables are still updated in two sequential turns, since all the y_i updates occur *in parallel.*

The splitting schemes we will discuss are *primal-dual* optimization methods, *i.e.*, they provide both a primal and a dual solution to problem (P). In addition, the construction of the schemes is mainly motivated from the dual form of (P). In the following sections, we first derive the dual problem and then set the foundations for the derivation of the splitting methods by means of the proximal operator.

2.1 The dual problem

We start by deriving the *Lagrangian* for (P), which can be written as

$$\mathcal{L}(x, y; \lambda) = f(z) + g(y) + \langle \lambda, Lz + l - y \rangle \ , \tag{L}$$

where $\lambda = (\lambda_1, \ldots, \lambda_M)$, $\lambda_i \in \mathbb{R}^{p_i}$ are dual variables associated with the equality constraints introduced above. We use the concatenated variables when it comes to derivations for lighter notation. We make the following standing assumtption:

Assumption 2. The Lagrangian (L) associated to (P) has a saddle point, *i.e.*,

$$\mathcal{L}(z^*, y^*; \lambda) \leq \mathcal{L}(z^*, y^*; \lambda^*) \leq \mathcal{L}(z, y; \lambda^*) \quad \forall z, y, \lambda \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^p \ .$$

The dual problem of (P) can be derived by means of the Lagrangian formulation. We have that

$$\begin{aligned} d(\lambda) &= \min_{z,y} \{f(z) + g(y) + \langle \lambda, Lz + l - y \rangle\} \\ &= -\max_z \left\{-f(z) - \langle L^T\lambda, z\rangle\right\} - \max_y \{-g(y) + \langle y, \lambda\rangle\} + \langle l, \lambda\rangle. \end{aligned}$$

Making use of the convex conjugate function (see Appendix A), the dual function can be expressed as

$$d(\lambda) = -f^\star(-L^T\lambda) + \langle l, \lambda\rangle - g^\star(\lambda), \tag{D}$$

where $g^\star(\lambda) = \sum_{i=1}^{M} g_i^\star(\lambda_i)$ and $f^\star$ is the conjugate of the sum of the two functions, given by $(h + \delta_{\mathcal{D}})^\star(-L^T\lambda) = \min_\mu \left\{h^\star(-L^T\lambda - \mu) + \delta_{\mathcal{D}}(\mu)\right\}$ [3, Proposition 15.2]. The dual problem to solve becomes

$$\lambda^* = \operatorname*{argmin}_\lambda \ F(\lambda) + g^\star(\lambda) \ , \tag{SolveD}$$

where $F(\lambda) := f^\star(-L^T\lambda) - \langle \lambda, l\rangle$.

In the algorithms considered here, both formulations (P) and (D) play a significant role. Although we mostly deal with the primal problem (P), the dual counterpart is essential for drawing a complete picture of the relations between the several methods. The derivation of the algorithms is a result of the application of several simple iterative schemes in order to minimize (D). These schemes are based on the application of the proximal operator.

2.2 Proximal methods

For $f \in \Gamma_0(\mathbb{R}^n)$, its *proximal operator* $\mathbf{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\mathbf{prox}_f(x) := \operatorname*{argmin}_z \left\{f(z) + (1/2)\|z - x\|^2\right\} \ .$$

The operator is evaluated at a given point x and looks for a minimizer that makes a compromise between the minimizer of the function f and the point x.

We refer to proximal methods as being a family of abstract algorithmic schemes that find a minimizer of a (sum of) convex function(s) by means of the proximal operator. More details can be found in the recent monograph [72]. The course notes [89] also provide a detailed reference to the topic.

Proximal Point Algorithm (PPA) PPA is mostly a conceptual scheme for minimizing the function f described above by means of the **prox** operator. It is written as the iteration

$$z^{k+1} := \mathbf{prox}_{\rho^k f}(z^k) \ , \tag{2.2}$$

i.e., it minimizes f while not moving too far away from the previous minimizer. The distance to z^k is controlled by the sequence $\{\rho^k\}$. The algorithm converges for $\rho^k > 0$ and $\sum_{k=1}^{\infty} \rho^k < \infty$.

The method was introduced in a more general form than the one presented here in [81]. Convergence properties have been further analyzed in [46]. Although it is applicable under mild assumptions, the proximal operator might be difficult to evaluate in the case of joint minimization of more than one functions.

Proximal Gradient Method (PGM) Consider the case that we want to minimize $l(z) = f(z) + g(z)$, $f \in \Gamma_0(\mathbb{R}^n)$ is *differentiable* and $g \in \Gamma_0(\mathbb{R}^n)$. The proximal gradient method is the iteration

$$z^{k+1} := \mathbf{prox}_{\rho^k g}\left(z^k - \rho^k \nabla f(z^k)\right) \ . \tag{2.3}$$

The algorithm converges for $\rho^k > 0$, either constant or determined by line search methods.

It is interesting to rewrite the method as

$$\begin{aligned} z^{k+1} &= \operatorname*{argmin}_z \left\{ g(z) + (1/2)\|z - (z^k - \rho^k \nabla f(z^k))\|^2 \right\} \\ &= \operatorname*{argmin}_z \left\{ g(z) + f(z^k) + \langle \nabla f(z^k), z - z^k \rangle + (1/2\rho^k)\|z - z^k\|_2^2 \right\} \ . \end{aligned}$$

Consequently, the method minimizes the sum of the (possibly) nonsmooth function g and a *quadratic approximation* of the smooth function f centered at the previously computed optimizer z^k. If the stepsize is chosen to be fixed in the range $(0, 1/L_f]$, where L_f is a Lipschitz constant for ∇f, then the quadratic model upper bounds f around z^k and the method can be shown to converge [4]. In reality the method converges for any $\rho \in (0, 2/L_f)$, but the quadratic model does not necessarily act as an upper bound for stepsizes that are larger than $1/L_f$. Note that, under the extra assumption of smoothness of f, the algorithm can deal with the sum of two functions in the objective, in contrast to the PPA.

Douglas-Rachford Method (DRM) The DRM generalizes the decomposition property of PGM while dropping the assumption for smoothness of f. The algorithm is composed of a sequence of proximal steps, expressed as

$$\begin{aligned} v^{k+1} &= \mathbf{prox}_{\rho f}\left(\lambda^k - w^k\right) \\ \lambda^{k+1} &= \mathbf{prox}_{\rho g}\left(v^{k+1} + w^k\right) \\ w^{k+1} &= w^k + v^{k+1} - \lambda^{k+1} \end{aligned} \tag{2.4}$$

Here $\rho > 0$ is a stepsize, although the algorithm generalizes to variable stepsizes, as we will see later. The original motivation for the method was the decomposition of the 'difficult' proximal evaluation $\mathbf{prox}_{f+g}(x)$ of the PPA. The scheme dates back in the 50's [40], [35], while a more recent analysis in the framework of convex optimization is performed in [29].

2.3 Origin of the methods and a unified framework

We consider three approaches for solving (P), all based on decomposing the problem into simpler ones with respect to the f and g functions. A common trait of the resulting algorithms is that they are derived by applying the proximal methods presented in the previous section to the dual function (D).

Alternating minimization algorithm (AMA) [85] AMA derives from applying PGM to the dual problem (D), when considering $f(\lambda) = f^\star(-L^T\lambda) - \langle l, \lambda\rangle$ and $g(\lambda) = g^\star(\lambda)$ in (2.3). The first step involves the evaluation of the gradient of the dual function, while the second step uses this gradient to progress along its negative direction in order to minimize the negative dual function $-d(\lambda)$. The equivalence of AMA with PGM applied to the dual problem is derived in Appendix B.1. As we mentioned in the previous section, PGM can only be applied under the existence of a smooth function in the objective. In the dual problem, smoothness holds for $f^\star(-L^T\lambda)-\langle l,\lambda\rangle$ if and only if $f(z)$ in the original formulation is *strongly convex* (see Appendix A, Lemma A.3). Finally, since convergence of the PGM comes under stepsize restrictions, AMA converges for $0 < \rho < \frac{2\sigma_f}{\|L\|^2}$, for f being σ_f-strongly convex. The upper bound imposed on the stepsize is the inverse Lipschitz constant of the gradient of $f^\star(-L^T\lambda)$, which coincides with the inverse matrix norm of the Hessian of this function.

Algorithm 1 Alternating minimization algorithm (AMA)

```
Require: Initialize $\lambda^0 \in \mathbb{R}^p$, and $0 < \rho < \frac{2\sigma_f}{\|L\|^2}$
  loop
    1: $z^{k+1} = \operatorname*{argmin}_z \quad f(z) + \sum_{i=1}^M \langle \lambda_i^k, L_i z\rangle$
    2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho}g_i}\left(L_i z^{k+1} + l_i + \lambda_i^k/\rho\right),\ i = 1,\dots,M$
    3: $\lambda_i^{k+1} = \lambda_i^k + \rho(L_i z^{k+1} + l_i - y_i^{k+1}),\ i = 1,\dots,M$
  end loop
```

From the perspective of the Lagrangian function, the first step of AMA is equivalent to the mininimization of (L) with respect to the z variable. The second step involves the minimization of the *augmented Lagrangian (AL)*, that can be expressed for problem (P) as

$$\mathcal{L}_\rho(z,y;\lambda) = f(z) + g(y) + \langle \lambda, Lz + l - y\rangle + (\rho/2)\|Lz + l - y\|_2^2 \ . \quad \text{(AL)}$$

Augmented Lagrangian functions have a long history in the optimization literature [80], [8]. Roughly speaking, minimization of the augmented Lagrangian function instead of the classical one results to faster convergence due to better regulation of the problem through the quadratic term. The augmented Lagrangian minimization problem results to proximal steps that can be implemented in parallel. In the end, a dual multiplier update ensures convergence of the algorithm by enforcing consensus of the affine sequence of updates $\{Lz^k + l\}$ to $\{y^k\}$.

Remark 2.1. Notice that the form that f takes can be flexible, as long as the strong convexity assumption is satisfied. In the control framework we are interested, and assuming a quadratic cost in terms of both inputs and outputs, we can distinguish two possible formulations:

1. The problem is written in terms of the control inputs. In this case, f is strongly convex and the dynamics equation $Az = b$ vanishes. In this case, $f(z) = z^T Hz + r^T z$, for some new H, r and the stepsize is upper bounded by $\lambda_{\min}(H)/\|L\|^2$.
2. We have that $z^T Qz > 0$ for $z \neq 0$ and $Az = 0$, *i.e.*, positive definiteness of Q in the nullspace of A (see [39, Proposition 33]). In this case we can write the KKT system

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} = \begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}^{-1} .$$

   The KKT smatrix in nonsingular, and hence the first step of AMA can be solved by means of a matrix inversion. The stepsize is upper bounded by $\lambda_{\min}(K_{11})/\|L\|^2$.

Primal-Dual Algorithm (PDA) In the context of large-scale convex optimization, the evaluation of the minimizer of f, as it appears, *e.g.*, in the first step of AMA, might be undesirably expensive. This is, *e.g.*, the case when f is a quadratic function with a dense Hessian, the minimization of which would require an inversion. This motivated the development of numerous algorithms that constitute of a sequence of evaluations of proximal operators, where the gradients and linear operators involved in the steps are called explicitely without inversion. The way to achieve this is *linearization of* f, *i.e.*, the algorithms make use of a quadratic approximant of the function instead of the original one. In this way, only the gradient of f is involved in the computations.

In the case where only two functions are involved in the objective, the most popular scheme of this type is *Chambolle and Pock's method* [16]. It was originally introduced in [92] and further analyzed in [16] and [49]. The algorithm was generalized to an arbitrary number of functions in the works [20] and [23], and even to versions that deal with inexactness in the evaluation of the proximal steps [88]. The versions come under many different names, of which we adopt the *Primal-Dual Algorithm (PDA)* to describe a method that is generic enough to encapsulate most of the existing ones. The proposed algorithm is in line with the recent scheme presented in [22].

Algorithm 2 Primal-Dual Algorithm (PDA)

```
Require: Initialize $\lambda^0 \in \mathbb{R}^p$, $z^0 \in \mathbb{R}^n$. Choose stepsizes $\tau, \rho > 0$ such
  that $\sqrt{\tau\rho}\sqrt{\sum_{i=1}^M \|L_i\|^2} < 1 - (L_h/2)\tau$
  loop
    1: $z^{k+1} = \mathbf{prox}_{\tau\delta_{\mathcal{D}}}\left(z^k - \tau(Qz^k + c + L^T\lambda^k)\right)$
    2: $\lambda_i^{k+1} = \mathbf{prox}_{\rho_i g_i^\star}\left(\lambda_i^k + \rho_i(L_i(2z^{k+1} - z^k) + l_i)\right)$, $i = 1, \ldots, M$
  end loop
```

An important common characteristic of these methods is that they make use of the information that f is the sum of a smooth and a non-smooth term, h and $\delta_{\mathcal{D}}$, respectively, as presented in (2.1). Consequently, a quadratic model is constructed for f, *i.e.*, $\hat{f}(z) = g(z) + h(z^k) + \langle \nabla h(z^k), z - z^k \rangle + (1/2\tau)\|z - z^k\|_2^2$. If h is smooth, information about the Lipschitz constant of its gradient is incorporated into the algorithm, typically resulting to faster convergence. Finally, the PGM is applied to the dual function of $\hat{f}(z) + g(Lz + l)$, though with a different stepsize condition for convergence. The derivation is presented in Appendix B. Note that the difference from AMA is the fact that the latter works with the *exact* dual problem (D), while PDA works with the *dual of an approximation to the original* f. This already suggests that AMA is, generally, faster to converge in terms of iterations. The first step of the algorithm involves the projection of the evaluated gradient iteration onto the dynamics' subspace, while the second step constitutes of dual variable updates by means of proximal operators that can be performed in parallel, as with AMA.

A different interpretation for Algorithm 2 is that of a scheme which optimizes the *saddle function*

$$\mathcal{S}(z;\lambda) = \langle Lz + l, \lambda\rangle + h(z) + \delta_{\mathcal{D}}(z) - g^{\star}(\lambda) \ . \tag{S}$$

The optimization procedure takes place in two steps; a proximal step involving the dynamics and a *linear approximation* of h, followed by a proximal step that involves the conjugate function $g^{\star}$.

Alternating direction method of multiplier (ADMM) [40], [35], [41] This is probably the most popular of the splitting methods, mostly due to its simplicity and the very few assumptions for convergence in comparison to other splitting schemes. It is also known as *split Bregman method* [44]. In the case of ADMM, the functions $f \in \Gamma_0(\mathbb{R}^n)$ and $g \in \Gamma_0(\mathbb{R}^p)$. Application of the PPA to the dual function (D) results to the minimization of a composite augmented Lagrangian, commonly known as the *method of multipliers* [53],[77],[8]. The difficulty to deal with the composite proximal operator, as discussed earlier, motivated research for a scheme that can split the proximal step into two. ADMM can be derived as a special case of DRM applied to the dual problem. The derivation is presented in Appendix B.

Algorithm 3 Alternating direction method of multiplier (ADMM)

```
Require: Initialize $y^0 \in \mathbb{R}^p$, $\lambda^0 \in \mathbb{R}^p$, and $\rho > 0$
  loop
    1: $z^{k+1} = \operatorname*{argmin}_z \quad f(z) + \sum_{i=1}^{M}\langle \lambda_i^k, L_i z\rangle + (\rho/2)\sum_{i=1}^{M}\|L_i z + l_i - y_i^k\|^2$
    2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho}g_i}\left(L_i z^{k+1} + l_i + \lambda_i^k/\rho\right),\ i = 1,\dots,M$
    3: $\lambda_i^{k+1} = \lambda_i^k + \rho(L_i z^{k+1} + l_i - y_i^{k+1}),\ i = 1,\dots,M$
  end loop
```

Compared to AMA, ADMM only differs in the minimization of the augmented Lagrangian function in the first step. This trait has the advantage that *no stepsize restrictions* occur for ADMM, in constrast to

AMA and PDA. On the other hand, *the augmented Lagrangian minimization complicates the first step by the addition of a (possibly dense) quadratic form*, even in the case that the original structure of f allowed for a cheaper evaluation. This is not the case with AMA and PDA, where the first step remains simple.

2.4 Relaxation

Relaxation has been used speedup the convergence of PPA in [29], giving rise to the iteration

$$z^{k+1} = (1-\theta^k)z^k + \theta^k \,\mathbf{prox}_{\rho^k f}(z^k),\ \theta^k \in (0,2)\ .$$

The scheme has been observed to speedup the convergence of ADMM for values $\theta^k > 1$, where it can be used by substituting Lz^{k+1} with

$$\theta^k L z^{k+1} - (1-\theta^k)(-y^k + l)\ .$$

The sequence $\{z^k\}$ is also over-relaxed in PDA where $\theta^k = 2$ is used at every iteration.

2.5 Termination

Given that problem (P) is convex, necessary and sufficient conditions for convergence are the *primal* and *dual feasibility conditions* (see, *e.g.*, [12, Chapter 5]). Writing the conditions for formulations (L) and (S), we get termination criteria for ADMM, AMA and PDA, respectively. The optimality conditions practically translate to primal and dual residuals that (asymptotically) go to zero as the algorithmic schemes progress. In practice, the algorithms are terminated when the conditions are satisfied to some prespecified accuracy. In Appendix C we present in detail how the derivation is performed for each of the algorithms.

We should note that, traditionally, first order (splitting) methods do not provide information about the feasibility of the problem. In the case of infeasibility, the subproblems to which the original problem is split will not converge to a common solution, something that is reflected in the residuals that do not decrease. After having observed this behavior for some time, the user can terminate the algorithm and claim infeasibility. It is only recently that the authors of [70] suggested an ADMM scheme that also returns a feasibility certificate. The idea is to use *homogeneous self-dual embedding*, a method commonly used with interior-point methods [91], [90]. The original problem is written as a feasibility problem by embedding the KKT conditions into a system of linear equations. From the solution of the embedding problem, one can either recover the solution to the original one, or a certificate for primal or dual infeasibility. Infeasibility detection using ADMM to solve QPs is also treated in the recent work [78]. In this manuscript we do not consider these variants, which should be used if infeasibility detection is crucial for the problem at hand.

3

Convergence results and accelerated variants

Slow convergence is usually the case with first-order methods, and the algorithms presented above are no exception to this rule. Slow convergence can be caused both by the primal and dual iterations. Indeed, the proximal step usually amounts to a projected gradient iteration, while the dual update is a gradient update step. In this aspect, the algorithms presented here cannot achieve a rate better than the existing lower complexity bounds for first-order methods [60], [62].

In what follows we frequently refer to convergence *in function values* and *in sequence values*. By saying that 'we have $O(1/k^q),\ q \in (0,2]$ global rate of convergence in *function values* for some function f', we mean that

$$f(z^k) - p^* \leq \frac{M}{k^q} \ ,$$

where p^* is the optimal value of f and $M > 0$. Accordingly, 'global $O(1/k^q)$ convergence rate *of a sequence* $\{z^k\}$' means that

$$\|z^k - z^*\| \leq \frac{M}{k^q} \ ,$$

where z^* is the optimizer and $M > 0$. Finally, 'ergodic convergence rate of $O(1/N^q)$ for a sequence $\{z^k\}$' (or in the function values of f)

means that, for $\bar{z}^N = \frac{1}{N}\sum_{k=1}^{N} z^k$,

$$\|\bar{z}^N - z^*\| \leq \frac{M}{N^q}, \quad f(\bar{z}^N) - p^* \leq \frac{M}{N^q} \ ,$$

respectively.

As was shown in the previous chapter, the splitting methods we discuss derive from the application of proximal algorithms to the dual composite function of (P). Consequently, the convergence results for these splitting methods also derive from existing convergence results of the proximal algorithms we use. Common sense dictates that the more knowledge we have about the function at hand, the better the convergence rates we can derive. Accordingly, *structural assumptions* are crucial for converging faster to optimality. The most important ones are given in Appendix A.

3.1 Sublinear convergence

First-order methods typically come with *sublinear convergence rates*, both in function values and in terms of the iterates. These rates are given from $q \in (0, 2]$ as presented in the previous section, depending on the structure of the problem at hand. Very roughly speaking, the following hold:

1. When *no structural assumptions on the functions are made*, sublinear convergence rates typically amount to an $O(1/k)$ global (or $O(1/N)$ ergodic) rate in function values.

2. Under the assumption of *smoothness*, several acceleration techniques can be applied, resulting to an $O(1/k^2)$ global rate in function values.

3. The convergence rate of the sequences of variables can usually be recovered as $O(1/\sqrt{k^q})$ from the function values' convergence rate.

4. Under the assumption of *smoothness and strong convexity*, linear convergence rates of the form $O(\omega^k),\ \omega \in (0, 1)$ can be recovered.

More specifically, ADMM converges at a global ergodic rate $O(1/N)$ in function values [82], [51] and at $O(1/\sqrt{k})$ in the sequences of primal and dual variables, for an arbitrarily large stepsize ρ [82]. Since AMA is equivalent to PGM applied to the dual problem (see Appendix B), an $O(1/k)$ convergence rate in the dual function values is proven in [4, Theorem 3.1]. In the case of PDA with two functions in the objective, a partial primal-dual gap function is shown to shrink with rate $O(1/N)$ for the ergodic sequences $\{\bar{z}^N\}$, $\{\bar{\lambda}^N\}$ and $\{\bar{y}^N\}$ in [16]. This result generalize to Algorithm 2.

3.2 Accelerated sublinear convergence

By making further assumptions on the function's structure, faster convergence rates can be recovered. In his work [75], Polyak proposed a way to accelerate the convergence of the gradient method, namely to use the modified update

$$\begin{aligned} \hat{z}^k &= z^k + \alpha^k (z^k - z^{k-1}) \\ z^{k+1} &= \hat{z}^k - \rho^k \nabla f(z^k) \ , \end{aligned}$$

on the smooth convex function f. This method is commonly known as the *heavy ball method*. In this way the next iterate depends on the last gradient update and the previous step $z^k - z^{k-1}$, which is called a *momentum sequence*. This seemingly small change of updating the new iterate as a linear combination of the two previous iterates greatly improves the performance of the original gradient scheme.

3.2.1 Nesterov acceleration

In his seminal paper [61], Nesterov modified the heavy ball method by simply evaluating the gradient at the extrapolated point $\hat{z}^k$ instead of z^k. In addition, he proposed a special formula for computing the relaxation sequence $\{\alpha^k\}$, resulting in an *optimal convergence rate for minimizing smooth convex functions using only gradient information.*

The simple update formula is:

$$\begin{aligned} \alpha^k &= \left(1 + \sqrt{4(\alpha^{k-1})^2 + 1}\right)/2 \\ \hat{z}^k &= z^k + \frac{\alpha^{k-1} - 1}{\alpha^k}(z^k - z^{k-1}) \\ z^{k+1} &= \hat{z}^k - \rho^k \nabla f(\hat{z}^k) \ , \end{aligned} \tag{3.1}$$

with $\alpha^0 = 1$. Subsequently, Güler extended Nesterov's results to the proximal point algorithm, handling the minimization of the sum of a smooth and a nonsmooth function [47]. More recently, Tseng [86] unified the analysis of fast PGM and proposed a condition for the acceleration sequence under which convergence is ensured. More specifically, the sequence $\{\alpha^k\}$ needs to satisfy

$$\frac{1 - \alpha^{k+1}}{(\alpha^{k+1})^2} \leq \frac{1}{(\alpha^k)^2} \ .$$

Both ADMM and AMA have benefited from accelerated relaxation schemes. Application of such scheme results in an $O(1/k^2)$ global rate of convergence in function values; a rate that is *optimal for first order methods involving a smooth and a nonsmooth function.* Convergence in terms of the sequences is based on the assumption of strong convexity. Roughly speaking, strong convexity associates the function values' difference to the sequences' difference (see definition of strong convexity in Appendix A), thus when the optimal $O(1/k^2)$ rate in terms of the primal (dual) function values is achieved, the primal (dual) sequences converge with rate $O(1/k)$ [43],[6],[16]. Apart from the theoretical results, the scheme has been observed to practically accelerate convergence in numerous problem instances. In addition, the extra computational cost is insignificant.

FAMA The accelerated version of AMA makes use of accelerated relaxation schemes on the dual sequence $\{\lambda^k\}$ [43]. For a smaller stepsize than this of the basic version (halfed), *convergence of the dual objective value* at rate $O(1/k^2)$ has been proven. In the same way that AMA is equivalent to PGM applied to the dual problem, FAMA is equivalent to the accelerated version of PGM, denoted as FPGM (Algorithm (11) in Appendix B). The algorithm was popularized under the name *Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)* [4], for the special case where $g(z) = \|z\|_1$. A rate of $O(1/k)$ of the primal sequence is also proven recently in [6]. FAMA can practically be applied to every problem that AMA can solve.

Algorithm 4 Fast alternating minimization algorithm (FAMA)

```
Require: Initialize $\alpha^0 = 1$, $\lambda^0 = \lambda^{-1} \in \mathbb{R}^p$, and $\rho \leq \frac{\sigma_f}{\|T\|^2}$
  loop
    1: $z^k = \operatorname*{argmin}_z \quad f(z) + \sum_{i=1}^{M} \langle \hat{\lambda}_i^k, L_i z \rangle$
    2: $y_i^k = \mathbf{prox}_{\frac{1}{\rho} g_i} \left( L_i z^k + l_i + \hat{\lambda}_i^k / \rho \right), \; i = 1, \ldots, M$
    3: $\lambda_i^k = \hat{\lambda}_i^k + \rho(L_i z^k + l_i - y_i^k), \; i = 1, \ldots, M$
    4: $\alpha^{k+1} = \left( 1 + \sqrt{4(\alpha^k)^2 + 1} \right) / 2$
    5: $\hat{\lambda}_i^{k+1} = \lambda_i^k + ((\alpha^k - 1)/\alpha^{k+1})(\lambda_i^k - \lambda_i^{k-1}), \; i = 1, \ldots, M$
  end loop
```

FADMM A fast version of ADMM (FADMM), based on Nesterov's acceleration, was first presented in [43]. The algorithm is presented below. In the case that f and all the functions g_i, $i = 1, \ldots, M$ are strongly convex, a global $O(1/k^2)$ rate is achieved in terms of dual function values. In the absence of assumptions no convergence rates can be proven. In addition, *convergence must be enforced by means of a restart rule*, as will be discussed in the next section.

3.2.2 Adaptive restart

It is frequently the case that the momentum sequence generated from acceleration schemes can result to oscillatory behaviour in the convergence of the function value to the optimal one. The term $\frac{\alpha^k - 1}{\alpha^{k+1}}$ in (3.1) is the amount of required momentum; if this amount is underestimated convergence is slower, while in the opposite case it causes a rippling behaviour, leading again to slower convergence.

The authors in [68] propose a restarting scheme, namely performing

Algorithm 5 Fast alternating direction method of multiplier (FADMM)

```
Require: Initialize $\hat{y}^0 = y^{-1} \in \mathbb{R}^p$, $\hat{\lambda}^0 = \lambda^{-1} \in \mathbb{R}^p$, $\rho > 0$, $\alpha^0 = 1$.
  loop
    1: $z^k = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^M \langle \hat{\lambda}_i^k, L_i z \rangle + \frac{\rho}{2} \sum_{i=1}^M \|L_i z + l_i - \hat{y}_i^k\|^2$
    2: $y_i^k = \mathbf{prox}_{\frac{1}{\rho} g_i} \left( L_i z^k + l_i + \hat{\lambda}_i^k / \rho \right), \ i = 1, \dots, M$
    3: $\lambda_i^k = \hat{\lambda}_i^k + \rho(L_i z^k + l_i - y_i^k), \ i = 1, \dots, M$
    4: $\alpha^{k+1} = \left(1 + \sqrt{4(\alpha^k)^2 + 1}\right)/2$
    5: $\hat{y}_i^{k+1} = y_i^k + ((\alpha^k - 1)/\alpha^{k+1})(y_i^k - y_i^{k-1}), \ i = 1, \dots, M$
    6: $\hat{\lambda}_i^{k+1} = \lambda_i^k + ((\alpha^k - 1)/\alpha^{k+1})(\lambda_i^k - \lambda_i^{k-1}), \ i = 1, \dots, M$
  end loop
```

a test every (few) iteration(s). Depending on the result of the test, the momentum term is set back to $\alpha^k = 1$ and the procedure is restarted. In order to demonstrate the scheme, consider again the accelerated version of the gradient method presented in (3.1). Two simple tests are provided; one ensuring that the function value keeps decreasing (restart if $f(z^{k+1}) > f(z^k)$) and the other one checking whether the new direction is pointing towards the negative gradient (restart if $\langle \nabla f(\hat{z}^{k+1}), z^{k+1} - z^k \rangle > 0$). Although a heuristic, the scheme significantly accelerated the convergence of gradient and proximal gradient schemes it was initially applied to. The generalization to ADMM and AMA followed shortly after.

Since AMA is equivalent to PGM, restart can be immediately applied to its fast version, *i.e.*, Algorithm 11 in Appendix B. In order to avoid function evaluations, the use of the gradient-based restart test is preferred. Since the gradient iteration for Algorithm 11 is the proximal step $\lambda^{k+1} = \mathbf{prox}_{\rho g^\star}\left(\hat{\lambda}^k + \rho(Lz^k + l)\right)$, we can view it as a *generalized gradient scheme* [68] of the form:

$$\lambda^{k+1} := \hat{\lambda}^k + \rho G(\hat{\lambda}^k),$$

where $G(\hat{\lambda}^k)$ is the generalized gradient at $\hat{\lambda}^k$ (note that z^k is a function of $\hat{\lambda}^k$ from Step 2 of Algorithm 11). Consequently, the gradient-based restart test can be expressed as $\langle -G(\hat{\lambda}^k), \lambda^{k+1} - \lambda^k \rangle > 0$, and can be applied right after Step 3 of Algorithm 4

```
4: if $\langle \hat{\lambda}^k - \lambda^k, \lambda^k - \lambda^{k-1} \rangle > 0$
5:     $\hat{\lambda}^k = \lambda^{k-1}$
6:     Repeat Steps 1,2,3 of Algorithm 4.
7: else
8:     Go to Step 4.
```

Note that the algorithm cannot cycle since the restart condition will not be satisfied once $\hat{\lambda}^k = \lambda^{k-1}$. In the recent work [37], the restarted version of Algorithm 11, and, consequently, the restarted FAMA, is proven to *preserve the optimal $O(1/k^2)$ convergence rate* in function values.

Along the same lines, restart schemes have also been applied to ADMM in [43]. In this case, acceleration is applied to both sequences $\{y^k\}$ and $\{\lambda^k\}$. In order to ensure convergence, the combined residual $c^k = \rho^{-1}\|\lambda^k - \hat{\lambda}^k\|^2 + \rho\|y^k - \hat{y}^k\|^2$ needs to vanish in the sense $\lim_{k\to\infty} c^k = 0$. The restart scheme can be plugged to Algorithm 5 after Step 4 and is presented below. The constant $\eta \in (0,1)$ corresponds to the decay rate of the residual term c^k.

```
5: $c^k = \rho^{-1}\|\lambda^k - \hat{\lambda}^k\|^2 + \rho\|y^k - \hat{y}^k\|^2$
6: if $c^k < \eta c^{k-1}$
7: $\hat{y}_i^{k+1} = y_i^k + ((\alpha^k - 1)/\alpha^{k+1})(y_i^k - y_i^{k-1}),\ i = 1, \dots, M$
8: $\hat{\lambda}_i^{k+1} = \lambda_i^k + ((\alpha^k - 1)/\alpha^{k+1})(\lambda_i^k - \lambda_i^{k-1}),\ i = 1, \dots, M$
9: else
10:     $\alpha^{k+1} = 1,\ \hat{y}^{k+1} = y^k,\ \hat{\lambda}^{k+1} = \lambda^k$
11:     $c^k \leftarrow \eta^{-1} c^{k-1}$
12: endif
```

Remark 3.1. It is possible that the restart scheme slows down the convergence of both FAMA and FADMM. Every time a restart happens, one full iteration of the algorithm goes to waste since the previous point is used instead of the over-relaxed one. In the worst-case scenario the algorithm will require *double the number of iterations* it would actually need if no restart was applied. Subsequently, restart methods will not necessarily improve on the fast versions of the two algorithms presented above.

3.2.3 Acceleration for PDA

PDA also comes with an accelerated variant. However, and despite the fact that the algorithm can be recovered from PGM, as shown in Appendix B, the idea behind its acceleration is quite different from Nesterov's, since it is based on varying stepsizes and a varying relaxation parameter. Explaining the scheme in full detail is beyond the scope of this work, however, we state the algorithm in the next paragraph for completeness.

FPDA The accelerated variant of PDA comes under the assumption that $f = h + \delta_{\mathcal{D}}$ is σ_f-strongly convex, denoted hereafter as F(ast)PDA (Algorithm 6). The acceleration is achieved by means of adaptive change of the primal and dual stepsizes τ and ρ_i. Furthermore, instead of taking a fixed momentum sequence $\{2(z^{k+1} - z^k)\}$ as in Algorithm 2, a *variable relaxation parameter* θ^k is introduced. The scheme results in a *global* $O(1/k)$ *convergence rate for the primal sequence* $\{z^k\}$, [11, Theorem 19]. A local $O(1/k^2)$ convergence in function values also applies, as proven in [16, Theorem 2] in the case that two functions comprise the objective of Algorithm 2.

3.2.4 Smoothing

It is evident by now that smoothness is the key property that allows for accelerated convergence in first-order splitting schemes. Since the three algorithms of interest are primal-dual methods, it comes as no surprise that strong convexity of the function f is needed in order to achieve the fast rate. This is a direct result of the celebrated dual relation between smoothness and strong convexity (see Lemma (A.2) in Appendix A).

Algorithm 6 Fast Primal-Dual Algorithm (FPDA)

```
Require: Initialize $\lambda^0 \in \mathbb{R}^p$, $z^0 \in \mathbb{R}^n$, $y^0 \in \mathbb{R}^m$. Choose stepsizes $\tau, \rho_1, \dots, \rho_M > 0$ such that $\tau < 2\sigma_f/L_h$ $\mu \geq L_h + 1$, $\tau^0 \left(\sum_{i=1}^M \rho_i^0 \|L_i\|^2\right) \leq \sqrt{1 + \tau^0(2\sigma_f - L_h\tau^0)/\mu}$ and $\theta^0 = 1/\sqrt{1 + \tau^0(2\sigma_f - L_h\tau^0)/\mu}$.
  loop
    1: $z^{k+1} = \mathbf{prox}_{(\tau^k/\mu)\delta_{\mathcal{D}}}\left(z^k - (\tau^k/\mu)(Qz^k + c + L^T\lambda^k)\right)$
    2: $\hat{z}^{k+1} = z^{k+1} + \theta^k(z^{k+1} - z^k)$
    3: $\lambda_i^{k+1} = \mathbf{prox}_{\rho^k g_i^\star}\left(\lambda_i^k + \rho_i^k(L_i\hat{z}^{k+1} + l_i)\right)$, $i = 1, \dots, M$
    4: $\tau^{k+1} = \theta^k\tau^k$, $\theta^{k+1} = 1/\sqrt{1 + \tau^0(2\sigma_f - L_h\tau^{k+1})/\mu}$,
       $\rho_i^{k+1} = \rho_i^k/\theta^{k+1}$, $i = 1, \dots, M$
  end loop
```

Subsequently, the proximal regularization of a convex function, attributed to Moreau, was extended by Nesterov in the work [63]. More specifically, consider the class of nonsmooth convex functions

$$q(x) = \max_{z \in \mathcal{Z}} \left\{\langle z, Ax\rangle - \phi(z)\right\} \quad ,$$

for some $x \in \mathbb{R}^n$, where $A \in \mathbb{R}^{m\times n}$ and $\mathcal{Z}$ is compact and convex. Nesterov defines a *proximity function* for the set $\mathcal{Z}$, denoted as D, as a σ_D-strongly convex and continuous function (over $\mathcal{Z}$), with $\mathcal{Z} \subseteq \mathbf{dom}\, D$. The proximity function measures the distance to the *prox center* $z_0 = \underset{z\in\mathcal{Z}}{\operatorname{argmin}}\ D(z)$. It holds that $D(z) \geq (1/2)\|z - z_0\|_2^2$, $\forall z \in \mathcal{Z}$. The smooth approximation of q is given by

$$q_\rho(x) = \max_{z \in \mathcal{Z}} \left\{\langle z, Ax\rangle - \phi(z) - \rho D(z)\right\} \quad .$$

As a result, q_ρ is $(\|A\|^2/\rho\sigma_D)$-smooth.

A useful discussion that provides an intuitive explanation for the method can be found in [64]. The convergence of the method is analyzed in [63]. We briefly mention here that, for any accelerated scheme, the original (nonsmooth) problem will converge with rate $O(1/k)$ in function values. Smoothing is extended to composite minimization problems in [65], as well as to more general proximity functions [5]. Application of smoothing has also gained attention in the MPC community [59], as well as in solving continuous-time optimal control problems [26].

3.3 Linear convergence

The linear rate's advantage over the sublinear one is that, at least theoretically, any accuracy level can be achieved. In practice, linearly convergent method can behave either very well or very badly since the slope of the linear rate can vary arbitrarily in the interval $(0, 1)$. Consequently, sublinear methods often behave better than linear ones, when convergence to a moderate accuracy is the purpose. *Linear convergence rates can be achieved by making use of the basic versions of the algorithms (Algorithms 1, 2, 3) with the extra assumption of strong convexity and smoothness of at least one of the functions in the objective.* In other words, linear convergence emerges from *the structural properties* of the functions, if these properties are there. This stands in stark contrast to the accelerated versions of the algorithms presented above that require no extra assumptions, achieving the acceleration only by means of the injected relaxation sequences and variable stepsizes.

Linear convergence of ADMM has been recently proven for several problem formulations. In [25], the authors prove *global linear* $O(1/\omega^k)$, $\omega > 0$ *convergence* of both ADMM and PADMM under a variety of scenarios in which at least f is strongly convex and smooth.

AMA also has a linearly convergent version under the extra assumption that *the dual function* $f^\star(-L^T\lambda)$ *is strongly convex in the variable* λ (see [85, Proposition 2]). In this scenario, *all primal and dual sequences* $\{z^k\}$ *and* $\{y^k\}, \{\lambda^k\}$ *converge to their optimizers linearly.* The assumption can be quite restrictive in control problems since L is (almost) always a tall matrix.

Chambolle and Pock suggest a third algorithm in their paper, *which achieves global linear convergence of both the primal and dual sequences* [16, Theorem 3]. Strong convexity of the functions f and $g^\star$ is required as well as knowledge of the convexity modula σ_f and $\sigma_{g^\star}$. Strong convexity of the conjugate function translates to smoothness of the original ones (g_i), an assumption that is highly unlike to hold in our problems of interest. The result in generalized in the case of $M > 1$ functions in the objective [11, Theorem 24].

In Table 3.1 we provide an up-to-date report of the existing convergence rates of the methods and their accelerated variants. Wherever a dash '-' appears, it means that there does not exist (or we are not aware of) such a result. Note that linear convergence always comes under additional assumptions. In some cases, there might be recent advancements that outperform the results presented here.

	Stepsize restrictions	Strong convexity	Decouples variables of linear constraints	Convergence in function values	Convergence in sequences
ADMM	no	no	no	ergodic $O(1/k)$ [51], [82]	$O(1/\sqrt{k})$ [82], $O(\omega^k)$ [25], [57], [36], [66]
AMA	yes	yes on f	yes	-	$O(1/\sqrt{k})$ on the primal [6], linear
PDA	yes	no	yes	ergodic $O(1/k)$ in partial primal-dual gap [16, Theorem 1], [88]	linear [16, Theorem 3],[11, Theorem 24]
FADMM	no	no	no	$O(1/k^2)$ on the dual under assumptions	-
FAMA	yes	yes on f	yes	$O(1/k^2)$ on the dual [43], [6]	$O(1/k)$ on the primal [6]
FPDA	yes	yes on f	yes	-	local $O(1/k)$ on the primal [16, Theorem 2], global in [11, Theorem 19]

Table 3.1

4 Stepsize selection and Preconditioning

In this chapter we discuss a crucial issue that affects every first order scheme, namely how the limited information provided by a first-order oracle can be optimally used in order to speedup the algorithmic progress. The discussion boils down to two topics: stepsize selection and conditioning of the problem. Although seemingly different, these two aspects are strongly related. We first explore how the stepsize can be selected for the three methods we have presented, moving from well-behaved functions with strong regularity properties to functions for which very little information is provided. We subsequently generalize the notion of the stepsize and show how to select the right metric, *i.e.*, the right space in which the problem should be solved. This is equivalent to preconditioning of the problem.

4.1 Stepsize

As in every first order method, the stepsize is crucial for the speed of the convergence. In augmented Lagrangian methods, where the stepsize (of the dual update) coincides with the penalty parameter and can be practically unbounded, the question of selecting a suitable one remains open in the general case. We first provide a few results that have to do with optimally choosing the stepsize under some assumptions on the problem structure. We then move to the more general (and practical) case where we do not know much about the problem structure and hence we adapt the stepsize according to the latest information we get from querying the function.

4.1.1 Optimal stepsize selection

The case where the first term of the composite objective in (2.1), f, is smooth and strongly convex, is well-studied and has provided with strong results of linear convergece, as was discussed in Section 3.3. Based on these results, the stepsize parameters can be tuned so that the corresponding convergence rates are maximized. These computations, however, require knowledge of both the strong convexity constant (σ_f) and the smoothness constant (L_f) of the function. These quantities are, generally, difficult to recover. Thus, the majority of the authors treat the case of QPs, where both constants are associated to the extreme eigenvalues of the Hessian of the dual function (D).

In the case of ADMM, the authors of [36] recover the optimal stepsize ρ, having first proven a linear rate of convergence for inequality form, strongly convex QPs, and under an additional assumption that the constraint matrix is either full row rank or invertible.These requirements are relaxed in the recent works [79] and [38].

We mentioned is Section 3.3 that Chambolle and Pock's scheme also achieves a linear convergence rate for the sequences, provided that we have two functions in the objective (*i.e.*, $M = 1$) and that both of them are strongly convex. As is the case with ADMM, this is achieved by picking the optimal values for the primal and dual stepsizes τ and ρ as they appear in Algorithm 2. The optimal stepsizes are again chosen based on the convexity modula. The result is generalized in the case of multiple functions g_i in [11].

4.1.2 Practical stepsize selection

In most cases the problem does not have a favorable structure and/or there is absence of information needed to optimally compute the stepsize. Thereby, we resort to more practical schemes in order to speedup the convergence. Two approaches have been mostly followed in literature to address this issue.

The first approach involves heuristic schemes that give rise to suboptimal fixed stepsizes. The procedures that are followed approximate the ones applied when the problem indeed has the desirable structural properties. Some knowledge of the geometry is still needed, restricting these methods mostly to QPs. Note that in model predictive control and the associated problems of interest the existence of a quadratic term in the objective is usual. Consequently, although problems of this form might be uncommon in general, we are interested in such structures and will return to them in the preconditioning section.

The second approach is adaptive updating of the stepsize(s) based on new information that becomes available as the algorithms iterate. This is commonly achieved by trying to guess the local structural properties of the involved functions either by direct function evaluations, or indirectly, via, *e.g.*, the residuals' evolution. As mentioned in Section 2.5, splitting algorithms converge when consensus to a common solution between the subproblems has been achieved. A good indication for this is the convergence of the primal and dual residuals r^k and s^k (Appendix C). Since the faster the residuals decrease the faster the termination of the algorithm, it intuitively makes sense to try and balance the residuals so that they converge at the same speed. Furthermore, computation of the residuals per iteration exhibits how they evolve, and thus it makes sense to try and 'correct' their behaviour if it is not desirable. This can be performed via the stepsizes.

AMA Although in the original version of AMA, as presented in [85], stepsize selection rules are not explicitely discussed, the equivalence of the method (and its accelerated version FAMA) with the PGM (and FPGM) algorithms [4], [6] allow for variable stepsizes. This is achieved by means of a *backtracking stepsize rule*. Backtracking is highly used in practical optimization for computing a suitable stepsize without having much knowledge about the structure of the problem [67, Chapter 3]. The approach was first developed in [4] for the ISTA and FISTA methods, and has as follows:

First, recall from (2.3) that AMA is PGM applied to the negative dual problem. In the same section we have shown how PGM can be interpreted as a proximal step involving a *quadratic approximation* of the smooth part (F) of the composite objective. Consequently, a quadratic model that upper bounds the negative dual function can be constructed at a given point μ, denoted as $-\hat{d}(\lambda) = Q_{L_F}(\lambda, \mu)$, where

$$Q_{L_f}(\lambda, \mu) := F(\mu) + \langle \lambda - \mu, \nabla F(\mu) \rangle + \frac{L_F}{2} \|\lambda - \mu\|^2 + g^{\star}(\lambda) \quad . \tag{4.1}$$

We denote the solution of (4.1) as

$$p_{L_f}(\mu) = \mathbf{prox}_{g^{\star}/L_F} \left(\mu - (1/L_F) \nabla F(\mu) \right) \quad .$$

A sufficient condition for ISTA to converge is that $-d(p_{L_F}(\mu)) \leq Q_{L_F}(p_{L_F}(\mu), \mu)$ [4, Lemma 2.3], *i.e.*, the original function evaluated at the next iterate is below the corresponding value of the quadratic model. Subsequently, an *estimate of the local Lipschitz constant* $\bar{L}_F$ can be computed at every iteration such that the condition is satisfied. The scheme can give rise to significantly smaller estimates of the Lipschitz constant compared to the conservative upper bound L_F, and, hence results to larger stepsizes $\rho^k = 1/\bar{L}_F^k$. The same backtracking rule applies to FISTA, and thus to the FAMA algorithm.

PDA Same as with ADMM, PDA's primal and dual residuals are inversely propotional to the (primal and dual) stepsizes τ and ρ_i, as indicated in (C.8). Consequently, one can achieve some residual balancing, and thus faster convergence, by adaptively choosing the stepsizes based on the latest residuals' update. The authors of [42] suggest checking a backtracking condition in order to guarantee convergence. In the case of PDA, convergence is based on ensuring that the primal and dual sequences move toward the solution set at every iteration, hence we have monotonicity of the sequences. The backtracking scheme reduces the stepsizes when this monotonic decrease is about to be violated. It is quite useful in practice since it often leads to long stepsizes that violate the stability conditions, exactly as with AMA. The full details can be found in [42].

ADMM Observing the ADMM iterates (Algorithm 3), one easily identifies that the dual stepsize or penalty parameter ρ is nothing but a weight on the penalization of the primal feasibility condition $y - Lz - l$, *i.e.*, a weight on the primal residual r^k (C.10). Consequently, by monitoring the primal residual one can either increase ρ when it gets big or decrease it when it is small compared to s^k. This is exactly the scheme proposed in [52], that reads as follows:

$$\rho^{k+1} = \begin{cases} 2\rho & \|r^k\| > c\|s^k\| \\ 0.5\rho & \|s^k\| > \|r^k\|/c, \\ \rho^k & \text{otherwise} \ , \end{cases} \tag{4.2}$$

with $c > 1$. The numbers suggested above are indicative and can be scaled according to the problem. This adaptive stepsize scheme is proven to converge and can frequently reduce the required number of iterations in practical applications.

4.2 Preconditioning

The most ponounced weakness of general first-order methods is their inability to efficiently deal with ill-conditioned problems. There are two ways to deal with bad conditioning: Either to a) choose a new coordinate system that adjusts better to the geometry of the problem or b) to cast the problem in a different form such that its geometry becomes more favorable, and then apply the algorithms to solve the recasted problem.

Since strong duality allows us to treat a problem in both its primal and dual forms, we have to option to alter the geometry of both spaces. Consider the original problem (P). *Primal preconditioning* is equivalent to left preconditioning of the primal variables, where *dual preconditioning* is equivalent to left preconditioning of the constraints.

Consider the scaling variables $z_p = Dz$, $y^d = Ey$, with $D \in \mathbb{R}^{n \times n}$ and $E \in \mathbb{R}^{p \times p}$, with $E = \mathbf{diag}(E_1, \dots, E_{M+1})$ and $E_i \in \mathbb{R}^{p_i \times p_i}$, $i = 1, \dots, M$. The sub(super)scripts p and d indicate scaled variables from the primal or from the dual preconditioning matrices. Problem (P) is then transformed to

$$\begin{array}{ll} \text{minimize} & f(D^{-1}z_p) + \sum_{i=1}^{M} g_i(E_i^{-1}y_i^d) \\ \text{subject to} & ELD^{-1}z_p + El = y^d \end{array} , \tag{4.3}$$

or, by setting $(\cdot)_p = D(\cdot)$, $(\cdot)^d = E(\cdot)$ and $(\cdot)_p^d = L(\cdot)D^{-1}$, the problem can be expressed in its preconditioned version as

$$\begin{array}{ll} \text{minimize} & f(D^{-1}z_p) + \sum_{i=1}^{M} g_i(E_i^{-1}y_i^d) \\ \text{subject to} & L_p^d z_p + l^d = y^d \end{array} , \tag{PrecP}$$

with variables z_p and y^d.

Remark 4.1. Although the use of the word ‘preconditioning’ generally refers to multiplication with *any* matrix, while ‘scaling’ refers to preconditioning with diagonal matrices, in this context we use both words to refer to any general preconditioner. If the matrix has some special structure, this is explicitly mentioned.

There are two points to consider before casting the original problem to (PrecP).

1. The preconditioned version should be faster to solve in terms of iterations.
2. The subproblems should not become computationally too costly to solve.

In practice, we look for a reformulation that is, in total, cheaper to solve than the original one. This requirement already poses serious restrictions on the structure of the scaling matrices D and E. Furthermore, the most important question that arises is *how to choose a good preconditioner*. As we will see, a common way is to optimize the known convergence rate of an algorithm equivalent to the splitting scheme of interest.

4.2.1 Preconditioners under structural assumptions

Same as with the stepsize, knowledge of the structure of the function f enables the computation of (in many cases optimal) preconditioners. In this section the standing assumption is *the existence of positive definite matrices M and H such that the first composite term of the negative dual function* (D)*, i.e., the function $F(\lambda) = f^\star(-L^T\lambda) - \langle l, \lambda\rangle$ is M_F-smooth and Σ_F-strongly convex.* Under this assumption, F can be lower and upper-bounded by quadratic functions. Manipulation of the level sets of the bounding functions so that they resemble those of F allows for faster convergence to the solution of the original problem.

AMA Applying AMA to problem (PrecP) we recover the preconditioned version of AMA, denoted as PrecAMA, which reads as follows:

Algorithm 7 Preconditioned Alternating Minimization Algorithm (PrecAMA)

```
Require: Initialize $\lambda^0 \in \mathbb{R}^p$, $\|ET\Sigma_f^{-1}T^TE^T\| = 1$.
  loop
    1: $z^{k+1} = \operatorname*{argmin}_z \quad f(z) + \sum_{i=1}^M \langle \lambda_i^k, L_i^d z\rangle$
    2: $(y_i^d)^{k+1} = E_i\,\mathbf{prox}_{g_i}^{P_i}\left(E_i^{-1}(L_i^d z^{k+1} + l_i^d + \lambda_i^k)\right),\ i = 1,\ldots,M$
    3: $\lambda_i^{k+1} = \lambda_i^k + L_i^d z^{k+1} + l_i^d - (y_i^d)^{k+1},\ i = 1,\ldots,M$
  end loop
```

There are several interesting observations to be made about Algorithm 7. Firstly, the stepsize ρ has been absorbed by the preconditioner. The initial restriction on the stepsize in the basic version of AMA is now translated into the condition involving the matrix norm, where $f(z)$ is Σ_f-strongly convex. Furthermore, the primal update in Step 1 can be expressed in the original variable z, since AMA is invariant when scaling the primal variables. The proximal step, however, changes significantly. In order to write it, we need to introduce the *generalized proximal operator* for a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, defined as

$$\mathbf{prox}_{\rho f}^{P}(x) := \inf_z \left\{ f(z) + \frac{1}{2\rho}\|z - x\|_P^2 \right\} \quad . \tag{4.4}$$

If P is not diagonal, the proximal step is not separable down to the component anymore, as is the case with several functions g_i (*e.g.*, indicator for nonnnegative orthant, l_1 norm). Consequently, the proximal step cannot be solved analytically. Even in the case of diagonal scaling, the closed form representation of the proximal step can be ruined. Table 4.1 in the end of the chapter illustrates some common cases of proximal operators and how diagonal scaling affects their representation. In Step 2 of the algorithm, the matrix P_i is defined as $P_i = E_i^T E_i$, hence a diagonal preconditioner E_i results to a diagonal P_i. Finally, note that the unscaled variable y does not appear in any of the updates, conveniently allowing us to work directly with the scaled version y^d without making any conversions.

Next, we discuss how to choose the matrix E. The motivation comes from the convergence properties of the PGM applied to the dual problem (D) (see Chapter 2 and Appendix B). The approach is discussed in detail in the recent work [39]. Starting from the equivalence of the two algorithms, the analysis is performed on the iteration $\lambda_i^{k+1} = \mathbf{prox}_{\rho g_i^\star}\left(\lambda_i^k + \rho(L_i z^{k+1} + l_i)\right)$ (Step 2, Algorithm 10). Note that this amounts to a simple proximal iteration in the direction of the negative gradient of F, and consider the stepsize to be $\rho = 1/L_F$. As mentioned before, this is equivalent to the minimization of the quadratic upper bound (4.1) of $-d(\lambda)$, $Q_L(\lambda, \mu) = F(\mu) + \langle \lambda - \mu, \nabla F(\mu)\rangle + \frac{L_F}{2}\|\lambda - \mu\|^2 + g^\star(\lambda)$. The quadratic model has uniform curvature L_h in all directions, being a poor approximation of the original composite dual function in the case that F is ill-conditioned. It is shown in [39, Proposition 29] that F is actually well approximated by the quadratic model $F(\mu) + \langle \lambda - \mu, \nabla F(\mu)\rangle + \frac{1}{2}\|\lambda - \mu\|_P^2$, where $P = L\Sigma_f^{-1}L$. In this way, the quadratic model is scaled to better approximate the level curves of F. This bound is tight for many functions, however it results to complicated proximal operators. It is thus preferable to approximate with some diagonal P such that $P^{-1} \approx L\Sigma_f^{-1}L$, and $P^{-1} \succ L\Sigma_f^{-1}L$ for guaranteeing convergence. Rewriting $P = E^T E$, the convergence condition is expressed as $I \succ EL\Sigma_f^{-1}L^T E^T$, while the quadratic model can be refined by minimizing its condition number, *i.e.*, $\frac{\lambda_{\max}(EL\Sigma_f^{-1}L^T E^T)}{\lambda_{\min}(EL\Sigma_f^{-1}L^T E^T)}$. The problem can be cast as an SDP, or solved by means of several heuristics [39, Section 6].

ADMM The scaled version of the algorithm reads as follows:

Algorithm 8 Preconditioned Alternating Direction Method of Multiplier (PrecADMM)

```
Require: Initialize $z^0 \in \mathbb{R}^p$, $\lambda^0 \in \mathbb{R}^p$, and $\rho > 0$
  loop
    1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \langle \lambda_i^k, L_i^d z \rangle + (\rho/2) \sum_{i=1}^{M} \| L_i^d z + l_i^d - (y_i^d)^k \|^2$
    2: $(y_i^d)^{k+1} = E_i \, \mathbf{prox}_{(1/\rho) g_i}^{P_i} \left( E_i^{-1} (L_i^d z^{k+1} + l_i^d + \lambda_i^k / \rho) \right), \quad i = 1, \ldots, M$
    3: $\lambda_i^{k+1} = \lambda_i^k + \rho (L_i^d z^{k+1} + l_i^d - (y_i^d)^{k+1}),\ i = 1, \ldots, M$
  end loop
```

The algorithm is quite similar to PrecAMA, except for the addition of the augmented Lagrangian term in the objective.

Same as with AMA, the purpose is to compute an optimal preconditioner E based on the properties of the dual function (D). Ignore for a moment the dynamics equation from the function $f(z)$, *i.e.*, resulting to $f(z) = \frac{1}{2} z^T Q z + c^T z$, M_f-smooth and Σ_f-strongly convex, and assume that the matrix L is full row rank. From the dual relation between smoothness and convexity (Lemma A.2), it holds that F is $\lambda_{\min}(L M_f^{-1} L^T)$-strongly convex and $\lambda_{\max}(L \Sigma_f^{-1} L^T)$-smooth. In addition, application of the Douglas-Rachford algorithm to (D) with the function F having these properties results to a linear convergence rate of the dual sequence that improves with the ratio $\frac{\lambda_{\max}(L \Sigma_f^{-1} L^T)}{\lambda_{\min}(L M_f^{-1} L^T)}$ [38, Proposition 6]. Note that the above expression corresponds to some notion of condition number, formulated as the ratio of the maximum eigenvalue of the quadratic upper bound to the minimum eigenvalue of the quadratic lower bound for the function F. Consider now the preconditioned version of the problem, *i.e.*, (PrecP), where only the constraints are scaled with E while the primal variables remain unscaled. The new condition number for the scaled problem becomes $\frac{\lambda_{\max}(E L \Sigma_f^{-1} L^T E^T)}{\lambda_{\min}(E L M_f^{-1} L^T E^T)}$. Consequently, we can freely choose the matrix E so that the ratio becomes one. However, choosing any positive definite matrix would result to a complicated proximal step (Step 2) in Algorithm 8, as discussed before. This is why E is typically set to a diagonal matrix. The matrix is commonly recovered by solving an SDP (see [36], [38]) or by means of (suboptimal) heuristic schemes if the SDP is too expensive to solve [39]. Note that the preconditioning problem is solved once and offline.

Unfortunately, the indicator function for the dynamics $\delta_{\mathcal{D}}$ results to loss of smoothness for f. Furthermore, in MPC problems we typically encounter L to be a tall, full column rank matrix. In these cases there are several heuristics for recovering a matrix E that does its best in scaling the problem given the restrictions. These schemes are analyzed in the works [36] and [38].

PDA In the case of the PDA, the philisophy behind the derivation of preconditioners is quite different from the one followed with AMA and ADMM. The results appearing in the literature derive mostly from the fact that PDA can be written as the application of the PPA (Chapter 2) to a variational inequality involving the optimality conditions of (S) [50]. However, preconditioning can be intuitively interpreted in the same way as before, *i.e.*, through the construction of quadratic approximations for the saddle function (S), followed by tuning of the curvature in different directions. The scaled version of the algorithm is:

Algorithm 9 Preconditioned Primal-Dual Algorithm (PrecPDA)

```
Require: Initialize $\lambda^0 \in \mathbb{R}^p$, $z^0 \in \mathbb{R}^n$. Choose $T, P$ such that $\sqrt{\sum_{i=1}^M \|P_i^{1/2} L T^{-1/2}\|^2} < 1 - (L_h/2)\|T\|$.
  loop
    1: $z^{k+1} = \mathbf{prox}_{\delta_{\mathcal{D}}}^{T^{-1}} \left( z^k - T(Qz^k + c + L^T(\lambda^d)^k) \right)$
    2: $(\lambda_i^d)^{k+1} = \mathbf{prox}_{g_i^\star}^{P_i^{-1}} \left( (\lambda_i^d)^k + P_i(L_i(2z^{k+1} - z^k) + l_i) \right)$, $i = 1, \dots, M$
  end loop
```

The matrices T and P are related to the preconditioners D and E via the equations $T = (D^T D)^{-1}$ and $P_i = E_i^T E_i,\ i = 1, \dots, M$.

The preconditioned version of PDA is a direct generalization of the work [74] with $M = 1$. The stepsizes τ and ρ of the original version

(Algorithm 2) are now picked to be different along the dimensions of both the primal and the dual problem. This can be easily shown if we take a close look at the iterations of Algorithm 9. The first iteration can be expressed as

$$z^{k+1} = \operatorname*{argmin}_{z} \quad \delta_{\mathcal{D}}(z) + \langle z, Qz^k + c + L^T(\lambda^d)^k \rangle + \frac{1}{2}\|z - z^k\|_{T^{-1}} \ .$$

For fixed λ^d, this step corresponds to the minimization of the convex part of the saddle function (S), using a linear approximation of h and regularized by a quadratic term with different curvature in each direction. Note that, in a manner similar to AMA, the minimization problem involves a quadratic upper bound of the original function. Picking the curvature (inverse stepsize) of the proximal term results in shaping the level sets of the original problem. Along the same lines, the second iteration

$$(\lambda_i^d)^{k+1} = \operatorname*{argmin}_{\lambda_i^d} \quad g_i^\star(\lambda^d) - \langle \lambda^d, L(2z^{k+1} - z^k) + l \rangle + \frac{1}{2}\|\lambda_i^d - (\lambda_i^d)^k\|_{P_i^{-1}}$$

involves the maximization of a quadratic lower bound of the concave part of the saddle function (S). The level curves are shaped through P_i.

Convergence of Algorithm 9 follows as long as the condition $\sqrt{\sum_{i=1}^M \|P_i^{1/2} L T^{-1/2}\|^2} < 1 - (L_h/2)\|T\|$ holds. This is a sufficient condition that proves convergence of PDA, initially derived in [88] and [23]. In the preconditioned version developed in [74], where $h = 0$, the condition becomes less conservative (allows for larger stepsizes).

4.3 General functions

In the previous section we focused on strongly convex and smooth functions f so that the smooth part of the optimization problem is nicely approximated by quadratic functions. In the general case, though, such a property might be missing, hence the techniques for deriving preconditioners that were discussed before are not applicable.

When this is the case, is has been practically observed that improving the conditioning of the L matrix speeds up the convergence of the corresponding algorithms [18], [34], [39], [74]. This is typically done by means of *equilibration*, *i.e.*, by choosing E and D^{-1} in (PrecP) so that the rows and columns of the equilibrated ELD^{-1} matrix have (approximately) the same norm. There is a variety of heuristic methods that perform equilibration, *e.g.*, [84], [15] and the impact on the convergence speed of the methods can be significant.

Remark 4.2. Since preconditioning only affects the data of the optimization problem, it is obvious that we can combine it with the acceleration techniques discussed previously without any further modifications. Consequently, both adaptive stepsize rules and Nesterov acceleration (with restart) can be blended into one algorithm that solves the preconditioned problem (PrecP). In the recent work [45], PGM is combined with all the above to give an improved version of the algorithm. Once applied to the dual problem (D), the proposed algorithm results to a preconditioned FAMA (with possibly variable stepsize).

Remark 4.3. A reasonable extension to the preconditioned versions of the algorithms would be to look for *variable metric schemes*, *i.e.*, preconditioners that adapt while iterating. Higher order methods emanate from the idea of variable preconditioners since exact (or approximate) Hessians of the problem at hand are evaluated while iterating, resulting in significantly faster convergence than ordinary first order methods. The trade-off comes again with the complexity of the proximal operator, as well as the computation of the preconditioner that has to be performed frequently. In addition, convergence of the algorithms becomes more subtle to prove. Few works exist that combine splitting algorithms with more sophisticated preconditioners such as [7], [21].

		Non-Preconditioned		Preconditioned	
		Fully Separable	Closed Form	Fully Separable	Closed Form
Orthant	Positive / Negative	✓	✓	✓	✓
Norm balls	l_∞	✓	✓	✓	✓
	l_1	✗	✗	✗	✗
	l_2	✗	✓	✗	✗
Norms	l_∞	✗	✗	✗	✗
	l_1	✓	✓	✓	✓
	l_2	✗	✓	✗	✗

Table 4.1: Properties of the generalized prox operator for a diagonal metric.

5

Numerical Linear Algebra

6

Examples

We demonstrate some of the methods presented in the previous sections with two optimal control problems. The first example involves MPC for tracking of a reference signal, boiling down to solving a sequence of QPs, while the second example considers the planetary soft landing problem, an originally nonconvex problem that is relaxed to an SOCP. We focus on explaining how to rewrite our problems so that we maximally exploit the ideas presented in Section **??**.

6.1 Aircraft control

In this example the linearized model of a Boeing 747-200 (B747) is considered [48]. The model has $n = 12$ states and $m = 17$ inputs and the aim is tracking of a reference signal $r(k)$ for three of the states. We discretize with sampling period $T_s = 0.2s$ and consider in total a signal of 115 setpoints. Firstly, a *steady state target calculator* computes a pair of setpoints $(\delta x_s(k), \delta u_s(k))$ for the aircraft, according to a desired reference signal. Subsequently, an MPC controller is tracking the delivered setpoint. The steady-states are generated by solving a strongly convex dense QP with $n + m = 29$ variables and bound constraints on the inputs [48, Section II,B]. The affine term in the objective is a function of $r(k)$, hence the optimization has to be performed as many times as is the length of the reference signal. The MPC problem is a simple quadratic one, with $Q \succeq 0$ and the same bound constraints on the inputs. The affine term is also time-varying since it is a function of the generated setpoints.

Steady state calculator The problem to solve is

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\theta_s^T H_s \theta_s - h_s(k)^T \theta_s \\ \text{subject to} & \theta_{min} \leq \theta_s \leq \theta_{max} \end{array} , \tag{6.1}$$

with variables $\theta_s \in \mathbb{R}^{n+m}$ and $H_s \succ 0$. Since the objective is strongly convex, we can use accelerated versions of the methods. To this end, FAMA and CPII are valid options, however, the dense structure of H_s would require a forward backward substitution at each iteration, something that can be avoided. We thus take the Cholesky factorization of H_s, *i.e.*, $H_s = LL^T$, L is lower triangular and invertible and perform a change of basis, $\tilde{\theta}_s = L^T \theta_s$. Now the problem can be reformulated as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s \\ \text{subject to} & C\tilde{\theta}_s \leq d \end{array} , \tag{6.2}$$

with variables $\tilde{\theta}_s \in \mathbb{R}^{n+m}$, $\tilde{h}_s(k) = L^{-1} h_s(k)$. The matrix-vector pair (C, d) describes the polytopic constraints that are now imposed in the place of the simple bound constraints that we had in (6.1). This is the price paid for eliminating the dense Hessian in the objective. By introducing a slack variable $y = C\tilde{\theta}_s - d$, $y \leq 0$, we can apply FAMA to the modified problem with $f(\tilde{\theta}_s) = \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s$, $l(y) = \delta_-(y)$, $T = C$, $t = -d$. For the stepsize we choose $\rho = 1/\lambda_{max}(C^T C)$.

As a second option, we use ADMM with the parameters tuned as in [36] in the same setting. This version achieves linear convergence rate by means of the optimal stepsize selection $\rho = 1/\sqrt{\lambda_{min}(CC^T)\lambda_{max}(CC^T)}$. In our case C is singular and so we consider the smallest nonzero eigenvalue.

Accordingly we can use CPII. Problem 6.2 can be written in a saddle point form as

$$\min_{\tilde{\theta}_s} \max_{\lambda} \left\{ \langle C\tilde{\theta}_s - d, \lambda \rangle + \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s - \delta_+(\lambda) \right\} \ ,$$

so we can use CPII with $Z = \mathbb{R}^{n+m}$, $g_i^\star(\lambda) = \delta_+(\lambda)$, T, t as defined above. Note that there are no equality constraints, hence there is no ν-update. We initialize the primal stepsize $\tau^0 = 100$ according to [16, Theorem 2].

We solve the problem 115 times with the affine term varying slightly from one iteration to the other. We terminate based on the residual decrease, with the accuracy threshold set to 10^{-3} for FAMA and CPII and 10^{-4} for ADMM (see Remark **??**). FAMA needs 495 iterations on average, with average time 0.85ms per solve, ADMM 194 iterations at 0.56ms per solve and CPII 1100 iterations at 4.9ms per solve. The solutions achieved are quite accurate, with a normed relative error ($\|\theta_s - \theta_s^\star\| / \|\theta_s^\star\|$) of $\approx 10^{-5}$ for all the methods, sumed over all 115 instances. The optimal stepsize selection renders ADMM clearly superior in this case.

MPC for tracking The MPC problem described in [48, Section II] can be written in the condensed form

$$\begin{array}{ll} \text{minimize} & \delta_{u_s}^T \delta_{u_s} + h(k)^T \delta_{u_s} \\ \text{subject to} & C\delta_{u_s} \le d \ , \end{array} \tag{6.3}$$

with variables $\delta_{u_s} \in \mathbb{R}^{Nm}$, after having changed the basis in the same way as before. We solve the problem for the following scenarios: $N = 5$, cold start, warm started at the primal and dual optima of the previous solve. The outputs are reported in Table 6.1. We use the same normed relative error $\|(x, u) - (x^\star, u^\star)\| / \|((x^\star, u^\star))\|$ to evaluate the quality of the solution. ADMM behaves significantly better than the other two methods in terms of iterations, but FAMA is faster overall in timings. With the number of variables increasing, the cost per iteration starts being more evident when using ADMM. We observe that warm starting makes a big difference in terms of iteration counts.

		ADMM	FAMA	CPII
$N = 5$	Av. No. Iters. Cold\Warm	1362 \548	2279 \778	1544\825
	Min.\Max. No. Iters. Warm	72\1504	83\5947	1\2111
	Av. Time Cold \Warm (ms)	46.90\19.82	42.74 \14.82	75.16 \40.53
	Relative error	1.61×10^{-4}	1.62×10^{-4}	1.61×10^{-4}
$N = 12$	Av. No. Iters. Cold\Warm	734\422	2999\1150	1721\1053
	Min.\Max. No. Iters. Warm	171\1079	167\10093	1\3389
	Av. Time Cold \Warm		274.54 \110.27	430.25 \266.75
	Relative error	4.7×10^{-5}	4.6×10^{-5}	4.6×10^{-5}

Table 6.1

6.2 The planetary soft landing problem

The problem presented in [2] regards the situation where an autonomous spacecraft lands on the surface of a planet by using thrusters, which produce a force vector that has both an upper and a nonzero lower bound on its magnitude. The control constraints are thus represented by a nonconvex set which has the form of a ring. This kind of constraints appears in a plethora of optimal control problems, but the case of planetary landing is of interest because, under some assumptions, the optimal trajectories of a relaxed version of the problem (where the nonconvex constraint set is replaced with a convex one) are also optimal for the original problem. Hence a lossless convexification can be achieved. The relaxed problem results in being an SOCP.

We first present the original problem formulation:

$$\begin{array}{ll} \text{minimize} & (x_0 - z_0)^T Q (x_0 - z_0) + \alpha \sum_{i=0}^{N-1} \|u_i\|_2 \\ \text{subject to} & x_{i+1} = Ax_i + Bu_i + Ew_i, \ i = 0, \ldots, N-1 \\ & x_N = z_f, \\ & \gamma |e_1^T x_i| \leq e_2^T x_i, \ i = 0, \ldots, N \\ & 1 \leq \|u_i\| \leq c, \ i = 0, \ldots, N-1 \ , \end{array} \tag{6.4}$$

with variables x_i, u_i and $\alpha \in \mathbb{R}_{++}$ a positive weight. Variable $x = (p_x, p_y, v_x, v_y)$ is the state of the $x - y$ position and the corresponding velocity coordinates of the spacecraft. The conic constraint correspondes to a *glide slope* constraint, ensuring that the spacecraft remains in a cone defined by a minimum slope angle. The nonconvex constraint on the inputs can be convexified by lifting, as discussed in [2, Section 3]. The relaxed problem can be written as

$$\begin{array}{ll} \text{minimize} & (x_0 - z_0)^T Q (x_0 - z_0) + \alpha \sum_{i=0}^{N-1} \sigma_i \\ \text{subject to} & x_{i+1} = Ax_i + Bu_i + Ew_i, \ i = 0, \ldots, N-1 \\ & x_N = z_f, \\ & \gamma |e_1^T x_i| \leq e_2^T x_i, \ i = 0, \ldots, N \\ & \|u_i\| \leq c, \ i = 0, \ldots, N-1 \\ & \|u_i\| \leq \sigma_i, \ i = 0, \ldots, N-1 \\ & \sigma_i \geq 1, \ i = 0, \ldots, N-1 \ , \end{array} \tag{6.5}$$

with variables x_i, u_i, σ_i.

There now exist several scenaria in which the solutions of (6.5) and (6.4) coincide. We consider one such scenario, similar to the numerical instances presented in [2, Section 5]. The data are

$$A = \begin{bmatrix} 0 & I \\ -\theta^2 I & \theta S \end{bmatrix} \in \mathbb{R}^n, \ B = E = \begin{bmatrix} 0 \\ I \end{bmatrix} \in \mathbb{R}^m \ , \ S = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} \ ,$$

$g = 3.7114$ is the gravitational acceleration of Mars, $w = -e_2 g$, $\theta = 1/(1.02595675 \times 24 \times 60 \times 60)$ is its rotation rate, $c = 4$, $\gamma = 1/\sqrt{3}$, $z_0 = (-15, 20, -10, 1)$, $z_f = (0, 0.1, 0, 0)$. The system is discretized using a zero-order hold with sampling period of $0.33s$. For a final time of $15s$ this gives $N = 45$. The weight matrix that penalized the initial state is chosen as $Q = \mathbf{diag}(2, 2, 1, 1)$.

Problem's (6.5) objective value does not exhibit any favorable characteristics (strong convexity or smoothness), and thus cannot benefit from an accelerated version of the methods mentioned above. In this case, we prefer to increase the number of slack variables and benefit from the simplicity of the prox operators. We first consider the augmented Lagrangian formulation AL with the following functions:

$$f(x, u, \sigma) = \left\{ (x_0 - z_0)^T Q (x_0 - z_0) + \sum_{i=0}^{N-1} \sigma_i \colon \boldsymbol{A}(x, u) = \boldsymbol{b} \right\} \tag{6.6}$$
$$g_i^1(Gx_i) = \delta_2(e_1^T x_i, (1/\gamma) e_2^T x_i),\ i = 0, \dots, N \tag{6.7}$$
$$g_i^2(u_i) = \delta_2(u_i, c),\ i = 0, \dots, N-1 \tag{6.8}$$
$$g_i^3(u_i, \sigma_i) = \delta_2(u_i, \sigma_i),\ i = 0, \dots, N-1 \tag{6.9}$$
$$g_i^4(\sigma_i) = \delta_+(\sigma_i - 1),\ i = 0, \dots, N-1\ . \tag{6.10}$$

We overloaded notation for the dynamics' equation by denoting

$$\boldsymbol{A} = \begin{bmatrix} -A & -B & I & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -A & -B & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & I & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -A & -B & I \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & I \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} Ew_0 \\ Ew_1 \\ \vdots \\ Ew_{N-1} \\ z_f \end{bmatrix},$$

and G is defined as $G = \begin{bmatrix} e_1^T \\ (1/\gamma) e_2^T \end{bmatrix}$. One can see that due to the several constraints, many copies of the variables have to be introduced. We denote the slack variables and the corresponding multipliers as

$$\begin{aligned} y_i = (y_i^1, y_i^2) = Gx_i, &\quad \text{with multiplier } \lambda_i^y = (\lambda_i^{y^1}, \lambda_i^{y^2}) \in \mathbb{R}^2 \text{ for (6.7).} \\ u_i = \tilde{u}_i, &\quad \text{with multiplier } \tilde{\lambda}_i^u \text{ for (6.8).} \\ (u_i, \sigma_i) = (\hat{u}_i, \hat{\sigma}_i), &\quad \text{with multiplier } \hat{\lambda}_i = (\hat{\lambda}_i^u, \hat{\lambda}_i^\sigma) \in \mathbb{R}^{m+1} \text{ for (6.9).} \\ \sigma_i = \tilde{\sigma}_i, &\quad \text{with multiplier } \tilde{\lambda}_i^\sigma \text{ for (6.10).}\ . \end{aligned}$$

The steps of ADMM as presented in Algorithm 3 are summarized below.

- The first step in the algorithm involves the minimization of (6.6). This optimization problem can be further split in two steps, a linearly constrained QP with variables (x, u) and an unconstrained QP for σ_i, decomposed for $i = 0, \dots, N-1$. The first QP can be expressed as

$$\begin{array}{ll} \text{minimize} & (1/2)s^T H s + h^T s \\ \text{subject to} & \boldsymbol{A}s = \boldsymbol{b}, \end{array} \tag{6.11}$$

over variable $s \in \mathbb{R}^{(N+1)n+Nm}$, where in the objective we group only quadratic and affine terms and the equality constraint corresponds to the dynamics of the system (see also [69]). We denote the following:

$$s = \begin{bmatrix} x_0 \\ u_0 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix}, \qquad h = \begin{bmatrix} -G^T(\rho y_0^k + (\lambda_0^y)^k) - 2Qz_0 \\ \rho(\tilde{u}_0^k + \hat{u}_0^k) + (\tilde{\lambda}_0^k + \hat{\lambda}_0^k) \\ \vdots \\ -G^T(\rho y_N^k + (\lambda_N^y)^k) \end{bmatrix},$$

$$H = \begin{bmatrix} 2Q + \rho G^T G & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2\rho I & 0 & \cdots & 0 & 0 \\ 0 & 0 & \rho G^T G & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2\rho I & 0 \\ 0 & 0 & 0 & \cdots & 0 & \rho G^T G \end{bmatrix},$$

and
The update is the solution of the linear system

$$\begin{bmatrix} H & \boldsymbol{A}^T \\ \boldsymbol{A} & 0 \end{bmatrix} \begin{bmatrix} s \\ \nu \end{bmatrix} = \begin{bmatrix} -h \\ \boldsymbol{b} \end{bmatrix}, \tag{6.12}$$

The σ-update has the closed form solution

$$\begin{aligned} \sigma_i^{k+1} = & \frac{1}{2}\left(\hat{\sigma}_i^k + \tilde{\sigma}_i^k + (1/\rho)((\hat{\lambda}_i^\sigma)^k + (\tilde{\lambda}_i^\sigma)^k - \alpha)\right), \\ & i = 0, \dots, N-1 \ . \end{aligned} \tag{6.13}$$

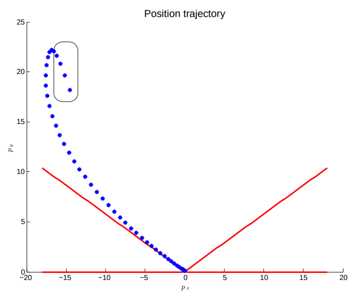
	ADMM	FADMM	aPDHG
No. Iters.	1224	1501	7598
Time (ms)			
Relative error	9.80×10^{-3}	10.70×10^{-3}	35.90×10^{-3}

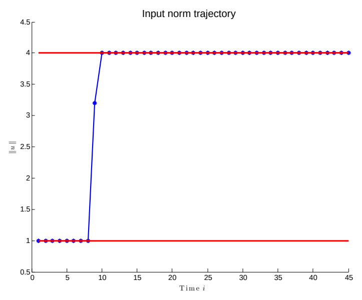
Table 6.2

- The second step invlolves a series of proximal minimization problems for each of the introduced variables (equations (6.7),(6.8),(6.9),(6.10). All the updates can be expressed in closed form and are separable across time. They involve projections on the nonnegative orthant, the second-order cone and the 2-norm ball. We will not write down the analytic expression for each one of the updates, but they can be easily calculated by consulting Table XXX

Another option is to go for the PDHG algorithm, which again has no assumptions on the problem structure. In this case we use the adaptive stepsize PDHG method (Algorithm **??**), which is known to work better in practice than its fixed stepsize counterpart. The steps are very similar to ADMM, with the only differences that 1) The dynamics enter the dual update ν (Step 4 of Algorithm **??**) and 2) All the proximal steps are performed with respect to the conjugate functions, as they are given in Table A.2.

Using over-relaxed ADMM and FADMM, the stepsize is chosen to be $\rho = 0.5$ and the over-relaxation parameter at 1.8. For the Adaptive PDHG we set $\tau = 100$. The error tolerances are set to 10^{-3}. The results are reported in Table 6.2. It is surprising that the over-relaxed ADMM bahaves better than the accelerated version in this case. We do not have a decisive argument on why this holds, since FADMM uses a heuristic rule for restarting which does not necessarily make it better in all cases. It seems that the small stepsize we use results to relatively smooth behaviour in the residuals, and hence the restarting scheme is not frequently activated. Furthermore, once the over-relaxation param-





(a) $x - y$ coordinates of spacecraft **(b)** Norm of inputs

Figure 6.1: Trajectory of the position as the spacecraft lands to the specified point $z_f = (0, 0.1)$. The initial state lies within some interval away from the desired one $z_0 = (-15, 20)$. With red color is depicted the glide angle constraint. Observe that many state constraints are active at the optimal trajectory. In the second plot the optimal input trajectory is depicted. As expected, the convex relaxation is exact, *i.e.*, the inputs stay in between 1 and 4. Note that almost all of them are saturated at optimality.

eter is set to 1, the number of iterations of ADMM increases. aPDHG converges very slowly, rendering it an unsuitable option in this case. We should mention that this is a specifically difficult problem to solve due to the big number of active constraints at optimality. This is the reason that even augmented Lagrangian-based methods, which usually behave well, need so many iterations for convergence.

6.3 Building economic control

In this section, we consider a control problem for a building heating system and design an MPC controller to minimize the total cost of operation that takes into account the prediction of the weather. The heating system is modeled by a discrete linear system

$$
\begin{aligned}
x_{i+1} &= Ax_i + Bu_i + Bd_i \\
y_i &= Cx_i \ .
\end{aligned}
$$

The system has ten states in x_t without specific physical interpretation. The system input $u_t = (u_1, u_2, u_3)$ includes the electrical power input (kW) to the heating system of the three zones. The disturbance input $d_t = (d_1, d_2, d_3)$ constitutes the outside temperature (°C), solar gains (kW), and internal gains (kW). The output of the system $y_t = (y_1, y_2, y_3)$ represents the temperature in the three building zones. There are ten states in the model, however they do not have any physical interpretation.XXX The sampling rate of the system is $T_s = 20$ min.

The purpose is to design an economic MPC controller with the objective is to keep the temperature of the building zones within the comfort constraints, while minimizing the energy bill. The economic MPC controller uses an economic linear cost function in Problem 6.14, as opposed to the usual quadratic cost function used in regulation problems. The inputs and the outputs of the model are subject to box-constraints.

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=0}^{N} c_i^T u_i \\
\text{subject to} & x_{i+1} = Ax_i + B_u u_i + B_d d_i, \ i = 0, \dots, N-1 \\
& y_i = Cx_i, \\
& u_{min} \le u_i \le u_{max} \ i = 0, \dots, N-1 \\
& y_{min} \le y_i \le y_{max}, \ i = 1, \dots, N \ ,
\end{array}
\tag{6.14}
$$

The input constraints capture the heating capacity of the building system with $u_{min} = 0 \ kW$ and $u_{max} = 15 \ kW$, while the output constraints ensure comfort in the building with $y_{min} = 22\,°\text{C}$ and $y_{max} = 26\,°\text{C}$. The sequence d_i denotes the disturbance prediction, and c_i is the electricity prices with a periodic high price and low price periods, i.e., $c_i = 0.04 \ \$/kWh$ (between 00:00Hrs to 10:00Hrs, and 16:00Hrs to 24:00Hrs) and $c_i = 0.2 \ \$/kWh$ (between 10:00Hrs to 16:00Hrs). The economic MPC problem can be formulated as a linear programming problem.

7

Summary

The first question that comes to mind when making use of a splitting method is how to perform the splitting. This choice can heavily affect the speed of the algorithm. Choosing a splitting pattern is equivalent to formulating the $M + 1$ subproblems that have to be solved in the three algorithmic schemes. Consequently, the choice will also confine the options for acceleration and preconditioning. A general guideline would be the following:

1. All subproblems should have a closed form solution if possible; if not, they should be cheap to solve. The whole purpose of using splitting on (P) is to end up with simpler subproblems.

All the acceleration techniques discussed in Chapters 3 and 4 can be practically seen as ways to robustify the algorithms against the stepsize parameter and the conditioning of the data. Note that, for a given set of data, there ideally exist theoretically optimal stepsize and relaxation constants. Due to the fact that these constants cannot be computed in most of the cases, we reduce the dependence on those by means of several heuristics. More specifically, we have observed that:

2. Preconditioning should always be used if possible. It has been em-

pirically observed that it is the most decisive factor for speeding up convergence.

3. Adaptation of the stepsize, especially in ADMM, can speedup convergence dramatically. In the case that simultaneous diagonalization can be used (see Chapter 5), adaptation should be used. Even if simultaneous diagonalization is not possible, it might be worth solving the first of a series of problems with adaptive stepsize strategy, despite the increased cost per iteration. The resulting stepsize (upon convergence) can then be used as a fixed one in the subsequent problems as the horizon recedes. It is observed that, even in the case that the optimal stepsize is known, adaptation might further reduce the number of iterations.

4. If an accelerated version of an algorithm can be used without heavily altering a well-structured problem, then it should be used. Acceleration generally improves significantly the number of iterations needed for convergence. Restarting of the scheme is optional in the case of AMA. Although restarting can end up increasing the number of iterations needed for convergence, in our experience it generally acts beneficially on the method. In both ADMM and AMA, the combination of the accelerated (restarted) versions with an adaptive stepsize strategy works well in practice. For FPDA the adaptation is already embedded in Step 4 of the method.

In many cases the approaches described above are competing. For example, one can precondition the problem so that an accelerated variant of a method can be used, but at the same time some favorable sparsity pattern of the original problem is lost. In our experience, there is no ‘golden rule’ when it comes to choosing a particular method and applying the various extensions for speeding it up. The choice of the method should be motivated from the problem’s structure and vice-versa. In the subsequent Figures 7.1, 7.2 and 7.3 we attempt to give a rough guideline on how an algorithm should be modified, once selected. The flow chart should be conceived as a proposed sequence of steps, in the sense that they usually (but not necessarily) enhance the performance of the algorithm. The steps suggested are also allowable, in the sense

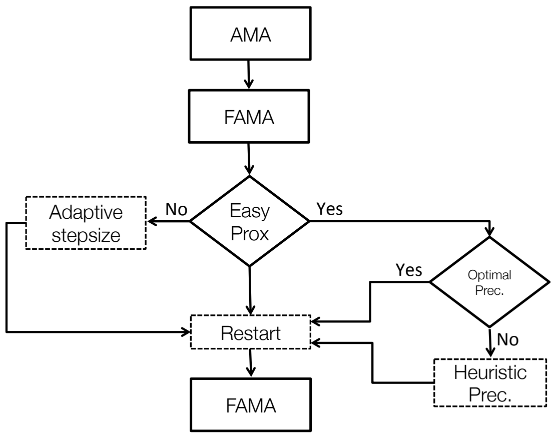


Figure 7.1: Flow chart for AMA.

that *convergence guarantees are ensured and the computational cost is not significantly increased*.

Warm-starting
Linear programs, ADMM for non-quadratics
Regarding the choice of the algorithm, there is typically a tradeoff between expensive operations and number of iterations. This tradeoff has to be evaluated on the specific application. For example, time-critical applications might tolerate an increased computational cost per iteration in order to achieve a (relatively) accurate solution in less iterations. Resource-limited applications might, on the other hand, prefer lighter computations at the expense of more iterations for convergence. It is difficult, however, to know in advance which of the two approaches will take less time. A good indicator is the average number of flops, *i.e.*, the number of operations per iteration multiplied by the average number of iterations for each algorithm.

Below, we examine each method separately, in more detail.

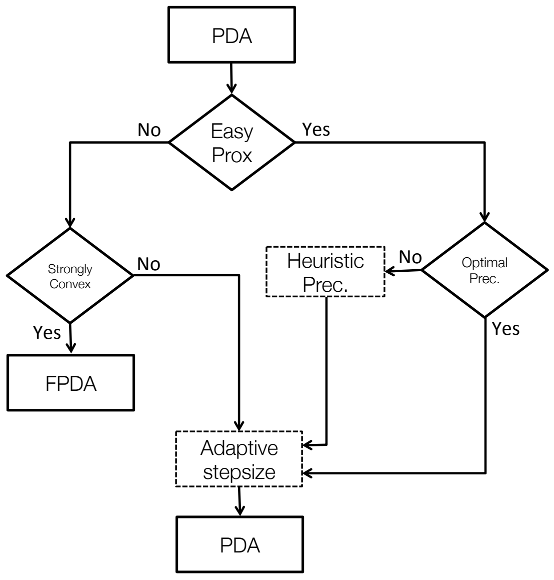


Figure 7.2: Flow chart for PDA.

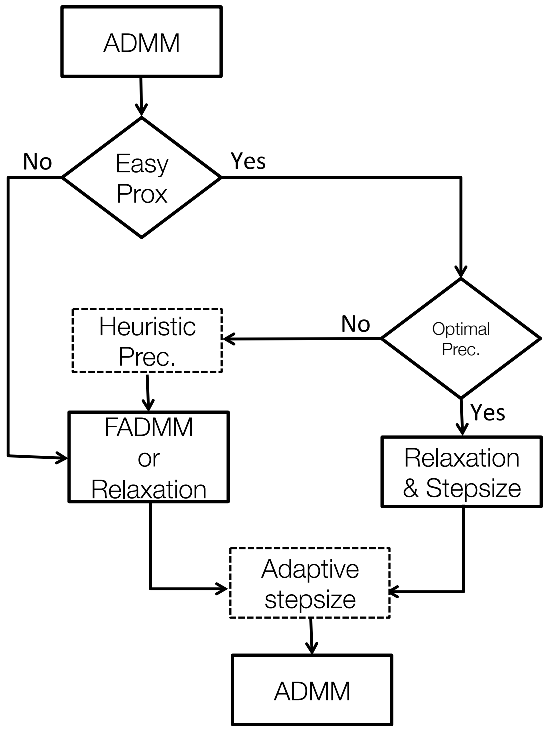


Figure 7.3: Flow chart for ADMM.

Appendices

A

Definitions

In this Appendix we give several definitions of notions that appear throughout the survey.

Subdifferential and Conjugacy

Definition A.1. (Subdifferential): The subdifferential of a convex function f at x is defined as

$$\partial f(x) = \{u : \langle u, z - x\rangle \leq f(z) - f(x)\ \forall z \in \mathbf{dom}\, f\}.$$

Note that when f is differentiable, $\partial f(x) = \{\nabla f(x)\}$.

Definition A.2. (Conjugate): The conjugate of a convex function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, denoted by $f^\star : \mathbb{R}^n \to \overline{\mathbb{R}}$, is defined as

$$f^\star(\lambda) = \sup_z \{\langle z, \lambda\rangle - f(z)\} \quad .$$

Theorem A.1. Let $f \in \Gamma_0(\mathbb{R}^n)$. The following relation holds:

$$u \in \partial f(x),\ \text{for some } x \in \mathbb{R}^n,\ \Leftrightarrow\ x \in \partial f^\star(u) \quad .$$

The notion of the conjugate function is quite useful in the framework of first-order methods for convex optimization. Below, we give some pairs of conjugate functions that we usually meet in the problems of interest.

Description	$f(z)$	$f^{\star}(\lambda)$
Nonnegative orthant	$\delta_{+}(z)$	$\delta_{-}(\lambda)$
Box/l_{∞}-norm ball ($\|z\|_{\infty} \leq l$)	$\delta_{\infty}(z, l)$	$l\|\lambda\|_1$
l_1-norm ball ($\sum_i \|z_i\| \leq l$)	$\delta_1(z, l)$	$l\|\lambda\|_{\infty}$
l_2-norm ball ($\|z\|_2 \leq l$)	$\delta_2(z, l)$	$l\|\lambda\|_2$
Lorentz cone ($\|z\|_2 \leq t$, $(z, t) \in \mathbb{R}^{n+1}$)	$\delta_2(z, t)$	$\delta_2(\lambda, y)$

Table A.2 by no means covers the wide spectrum of convex conjugate functions that can be analytically derived. We refer the interested reader to the work [19] for an exhaustive list of conjugate functions. Similarly, since proximal operators of convex functions have been derived in numerous works, we do not intend to list them here. We once more refer the interested reader to [19] for a list, as well as [72] for the derivation procedure of the most common ones. Finally, the Matlab library [71] implements many proximal operators for direct use, serving as a supplement to [72].

Structural properties

Definition A.3. (Smoothness): A function $f \in \Gamma_0(\mathbb{R}^n)$ is β-smooth if it is differentiable and

$$f(z) \leq f(x) + \langle \nabla f(x), z - x \rangle + (\beta/2)\|z - x\|^2 \ ,$$

holds for all $z, x \in \mathbb{R}^n$.

Definition A.4. (Strong convexity): A function $f \in \Gamma_0(\mathbb{R}^n)$ is β-strongly convex if

$$f(z) \geq f(x) + \langle u, z - x \rangle + (\beta/2)\|z - x\|^2 \ ,$$

holds for all $z, x \in \mathbb{R}^n$ and all $u \in \partial f(x)$.

Remark A.1. The above definitions generalize in the case of positive semi-definite matrices $P \in \mathbb{S}_{+}$ instead of scalars β. The inequalities then hold with $(1/2)\|z - x\|_P^2$.

Definition A.5. (Lipschitz continuous gradient): A differentiable function $f \in \Gamma_0(\mathbb{R}^n)$ has a β-Lipschitz continuous gradient if

$$\|\nabla f(z) - \nabla f(x)\| \leq \beta \|z - x\| \ .$$

β-Lipschitz continuous gradient is equivalent to β-smoothness of the function.

Lemma A.2. Let $f \in \Gamma_0(\mathbb{R}^n)$. The following statements are equivalent:
The function f is β-strongly convex function.
The conjugate function $f^\star \in \Gamma_0(\mathbb{R}^n)$ has $1/\beta$-Lipschitz continuous gradient.

Remark A.2. All the definitions and lemmata given above hold for $\mathbf{dom}\, f = \mathcal{Z} \subseteq \mathbb{R}^n$. We choose to restrict ourselves to $\mathbf{dom}\, f = \mathbb{R}^n$ for simplicity. In this manuscript, when the domain of a function is a subset of the real numbers, this is explicitly treated via indicator functions.

Useful identities Moreau identity is a very useful Lemma that associates a convex function with its conjugate. This is instrumental for deriving all the algorithms presented in this work, since, as we saw, their derivation depends on the application of proximal methods to the dual function (D), which is expressed via conjugate functions. Below we give Moreau identity associated to the proximal, as well as the generalized proximal operator (4.4).

Lemma A.3. Let $f \in \Gamma_0(\mathbb{R}^n)$. Then for any $x \in \mathbb{R}^n$

$$\mathbf{prox}_{\rho f^\star}(x) + \rho\, \mathbf{prox}_{f/\rho}(x/\rho) = x, \ \ \forall\, 0 < \rho < +\infty \ .$$

Lemma A.4. Let $f \in \Gamma_0(\mathbb{R}^n)$ and $P \in \mathbb{S}_{++}$. Then for any $x \in \mathbb{R}^n$

$$\mathbf{prox}^{P}_{\rho f}(x) + \rho P^{-1}\, \mathbf{prox}^{P^{-1}}_{f/\rho}(P(x/\rho)) = x, \ \ \forall\, 0 < \rho < +\infty \ .$$

B

Derivation of the algorithms from proximal methods

The subsequent results are instrumental as they demonstrate the equivalence of the AMA, PDA and ADMM to proximal algorithmic schemes, as presented in Chapter 2, when applied to the dual problem (D). We repeat here the dual function for clarity:

$$-d(\lambda) = F(\lambda) + g^\star(\lambda) \ ,$$

where $F(\lambda) := f^\star(-L^T\lambda) - \langle \lambda, l \rangle$.

AMA and PGM. The Alternating Minimization Algorithm can be derived from the application of the PGM iteration (2.3) to $-d$. The proof is inspired from [6, Lemma 3.2], where the equivalence between the accelerated versions of AMA and PGM is drawn.

Since f is strongly convex, it follows from Lemma A.2 that $f^\star$ is smooth. Subsequently, F is also smooth. Then ∇F is Lipschitz continuous with some constant L_F, and the proximal gradient algorithm reads

Algorithm 10 Dual Proximal Gradient Method

```
Require: Initialize $\rho < 2/L_F$.
  loop
    1: $z^{k+1} = \operatorname*{argmin}_z \quad f(z) + \langle L^T \lambda^k, z \rangle$
    2: $\lambda_i^{k+1} = \mathbf{prox}_{\rho g_i^\star}\left(\lambda_i^k + \rho(L_i z^{k+1} + l_i)\right),\ i = 1, \dots, M$
  end loop
```

The following Lemma holds:

Lemma B.1. The second step of Algorithm 10 is equivalent to steps 2 and 3 of Algorithm 1 combined, written as

$$\begin{aligned} y_i^{k+1} &= \mathbf{prox}_{g_i/\rho}\left(L_i z^{k+1} + l_i + \lambda_i^k/\rho\right) \\ \lambda_i^{k+1} &= \lambda_i^k + \rho(L_i z^{k+1} + l_i - y_i^{k+1}) \ . \end{aligned}$$

Proof. Step 1 of Algorithm 10 amounts to computing the gradient of $f^\star(-L^T\lambda) - \langle \lambda, l \rangle$. Using Theorem A.1,

$$-L^T \lambda^k \in \partial f(z^{k+1}) \Leftrightarrow z^{k+1} = \nabla f^\star(-L^T \lambda^k) \ .$$

Thus, the gradient of F can be expressed as

$$\nabla F(\lambda^k) = -L z^{k+1} - l \ .$$

Step 2 is, apparently, the proximal step with respect to $g_i^\star$ in the direction of the negative gradient.

We are now ready to prove the assertion. To this end, we first denote $\mu_i^k = \lambda_i^k + \rho(L_i z^{k+1} + l_i)$, and subsequently substitute the result of the proximal step y_i^{k+1} to the dual update λ_i^{k+1} :

$$\begin{aligned} \lambda_i^{k+1} &= \lambda_i^k + \rho(L_i z^{k+1} + l_i) - \rho\, \mathbf{prox}_{g_i/\rho}(\lambda_i^k/\rho + L_i z^{k+1} + l_i) \\ &= \mu_i^k - \rho\, \mathbf{prox}_{g_i/\rho}(\mu_i^k/\rho) \ . \end{aligned}$$

Using Moreau identity (A.3), it directly follows that

$$\lambda_i^{k+1} = \mu_i^k - \mu_i^k + \mathbf{prox}_{\rho g_i^\star}(\mu_i^k) = \mathbf{prox}_{\rho g_i^\star}(\mu_i^k) \ .$$

□

Finally, the accelerated version of PGM, named *Fast Proximal Gradient Method (FPGM)* [6],[39], applied to the dual problem, is given below. The method is the result of simply applying Nesterov's momentum scheme to Algorithm 10, as discussed in Chapter 3. Any other acceleration sequence [86] would result to similar algorithms.

Algorithm 11 Fast Dual Proximal Gradient Method

```
Require: Initialize $\rho \in (0, 1/L_F]$, $\lambda^{-1} = \lambda^0 \in \mathbb{R}^p$, $\alpha^{-1} = \alpha^0 = 1$.
  loop
    1: $\hat{\lambda}^k = \lambda^k + ((\alpha^{k-1} - 1)/\alpha^k)(\lambda^k - \lambda^{k-1})$
    2: $z^k = \underset{z}{\operatorname{argmin}} \quad f(z) + \langle L^T \hat{\lambda}^k, z \rangle$
    3: $\lambda_i^{k+1} = \mathbf{prox}_{\rho g_i^\star}\left(\hat{\lambda}_i^k + \rho(L_i z^k + l_i)\right)$, $i = 1, \ldots, M$
  end loop
```

PDA and PGM. Consider the problem (P) that can be written as

$$\text{minimize} \quad h(z) + \delta_{\mathcal{D}}(z) + g(Lz + l) \ ,$$

with $h(z)$, $\delta_{\mathcal{D}}(z)$ as defined in (2.1). We now consider the quadratic approximation of $h(z)$ expressed as:

$$\hat{h}(z) = h(x) + \langle \nabla h(x), z - x \rangle + (1/2\tau)\|z - x\|_2^2 \ ,$$

for all $z, x \in \mathbb{R}^n$ and $\tau > 0$. We have the following relations:

$$\min_z \left\{ h(x) + \langle \nabla h(x), z - x \rangle + (1/2\tau)\|z - x\|_2^2 + \delta_{\mathcal{D}}(z) + g(Lz + l) \right\} =$$

$$\max_\lambda \min_{z,y} \left\{ h(x) + \langle \nabla h(x), z - x \rangle + (1/2\tau)\|z - x\|_2^2 + \delta_{\mathcal{D}}(z) + g(y) + \langle \lambda, Lz + l - y \rangle \right\} =$$

$$\max_\lambda \min_z \ -\max_y \left\{ \delta_{\mathcal{D}}(z) + (1/2\tau)\|z - (\underbrace{x - \tau\nabla h(x)}_{v})\|_2^2 + \langle \lambda, Lz + l + y \rangle - g(y) \right\} =$$

$$\max_\lambda \min_z \left\{ \delta_{\mathcal{D}}(z) + (1/2\tau)\|z - v\|_2^2 + \langle \lambda, Lz + l \rangle + g^\star(\lambda) \right\} =$$

$$\max_\lambda \min_z \left\{ \delta_{\mathcal{D}}^r(z) + \langle L^T\lambda, z \rangle + \langle \lambda, l \rangle + g^\star(\lambda) \right\} =$$

$$\min_\lambda \left\{ (\delta_{\mathcal{D}}^r)^\star(-L^T\lambda) - \langle l, \lambda \rangle + g^\star(\lambda) \right\} \ ,$$

where we denoted the *regularized* dynamics indicator function as

$$\delta^r_{\mathcal{D}}(z) := \delta_{\mathcal{D}}(z) + (1/2\tau)\|z - x\|_2^2 \ .$$

Note that $\delta^r_{\mathcal{D}}$ is strongly convex, hence $(\delta^r_{\mathcal{D}})^\star$ is smooth. So we can apply the PGM to $-\hat{d}(\lambda) := (\delta^r_{\mathcal{D}})^\star(-L^T\lambda) - \langle l, \lambda\rangle + g^\star(\lambda)$. We can now follow the same reasoning as in the case of AMA.

Consider $x = z^k$ and $v^k = z^k - \tau\nabla h(z^k)$. Denote also the smooth part of $\hat{d}(\lambda)$ as $\hat{F}(\lambda) = (\delta^r_{\mathcal{D}})^\star(-L^T\lambda) - \langle l, \lambda\rangle$. Step 1 of Algorithm 2 can be expressed as

$$\begin{aligned} z^{k+1} &= \operatorname*{argmin}_z \left\{\delta_{\mathcal{D}}(z) + (1/2\tau)\|z - v^k\|_2^2 + \langle\lambda^k, Lz + l\rangle\right\} \\ &= \operatorname*{argmin}_z \left\{\delta^r_{\mathcal{D}}(z) + \langle\lambda^k, Lz + l\rangle\right\}. \end{aligned}$$

The optimality condition reads

$$\begin{aligned} -L^T\lambda^k &\in \partial\delta^r_{\mathcal{D}}(z^{k+1}) \\ z^{k+1} &= \nabla(\delta^r_{\mathcal{D}})^\star(-L^T\lambda) \ . \end{aligned}$$

Consequently, the gradient of $\hat{F}(\lambda)$ is

$$\nabla\hat{F}(\lambda) = -L\nabla(\delta^r_{\mathcal{D}})^\star(-L^T\lambda) = -Lz^{k+1} - l.$$

Finally, the proximal update of Algorithm 10 coincides with Step 2 of Algorithm 2, evaluated at the over-relaxed point $2z^{k+1} - z^k$. However, note that the convergence condition imposed on the stepsizes is not the same, *i.e.*, PDA needs $\sqrt{\tau\rho} < \frac{(1-\tau L_h/2)}{\sqrt{\sum_{i=1}^M \|L_i\|^2}}$. This condition already poses the restriction that $\tau < 2/L_h$. Consequently, PDA can be viewed through the PGM lens, although it cannot be fully explained through it. The details for the convergence of the algorithm are given in [23].

Remark B.1. The addition of the quadratic term to the indicator function results to strong convexification of the composite function $\delta^r_{\mathcal{D}}$. The implication of smoothness of the convex conjugate function $(\delta^r_{\mathcal{D}})^\star$ can also be interpreted in the framework of smoothing methods, as mentioned in Chapter 3.

ADMM and DRS. The DRS scheme constitutes of three iterations, and when applied to the dual problem results to:

$$v^{k+1} = \mathbf{prox}_{\rho F}\left(\lambda^k - w^k\right) \tag{B.1}$$

$$\lambda^{k+1} = \mathbf{prox}_{\rho g^\star}\left(v^{k+1} + w^k\right) \tag{B.2}$$

$$w^{k+1} = w^k + v^{k+1} - \lambda^{k+1} \ , \tag{B.3}$$

where $d = F + g^\star$, as defined in the begining of the chapter. The function F does not need to be smooth in this case. We are going to analyze the three iterations sequentially, following the approach form the lecture notes [89].

For (B.1), we have that

$$\mathbf{prox}_{\rho F}\left(\lambda^k - w^k\right) = \underset{v}{\operatorname{argmin}}\left\{F(v) + (1/2\rho)\|v - \lambda^k + w^k\|_2^2\right\} \ ,$$

the optimality condition of which is

$$0 \in -L\partial f^\star(-L^T v) - l + (1/\rho)(v - \lambda^k + w^k) \tag{B.4}$$

We are going to show that the proximal step (B.1) is equivalent to an augmented Lagrangian minimization update. For this purpose, consider the minimization problem

$$\text{minimize} \quad f(z) + (\rho/2)\|Lz + l + (\lambda^k - w^k)/\rho\|_2^2$$

with variable $z \in \mathbb{R}^n$, which can be equivalently written as

$$\begin{array}{ll} \text{minimize} & f(z) + (\rho/2)\|u\|_2^2 \\ \text{subject to} & Lz + l + (\lambda^k - w^k)/\rho = u, \end{array} \tag{B.5}$$

with variables $z \in \mathbb{R}^p, u \in \mathbb{R}^p$. Introducing a Lagrange multiplier $v \in \mathbb{R}^p$, the optimality conditions for problem (B.5) become:

$$-L^T v \in \partial f(z), \quad \rho u = v, \quad Lz + l + (\lambda^k - w^k)/\rho - u = 0 \ ,$$

which, by elimination of the variables z, u, can be written as

$$0 \in -L\partial f^\star(-L^T v) - l + (1/\rho)(v - \lambda^k + w^k),$$

which is (B.4).

Furthermore, we have from (B.1) that

$$\begin{aligned} v^{k+1} - \lambda^k + w^k &\in -\rho \partial F(v^{k+1}) \\ &= -\rho(-L\partial f^\star(-L^T v^{k+1}) - l) \\ &= \rho(Lz^{k+1} + l) \ , \end{aligned}$$

where z^{k+1} is a minimizer of problem (B.5). To wrap up, Step (B.1) can be written as

$$\begin{aligned} z^{k+1} &= \operatorname*{argmin}_z \left\{ f(z) + (\rho/2)\|Lz + l + (\lambda^k - w^k)/\rho\|_2^2 \right\} \\ v^{k+1} &= \lambda^k - w^k + \rho(Lz^{k+1} + l) \ . \end{aligned} \tag{B.6}$$

Step (B.2) reads as $\lambda^{k+1} = \mathbf{prox}_{\rho g^\star}\left(\lambda^k + \rho(Lz^{k+1} + l)\right)$. From Lemma (B.1), Step (B.2) is equivalent to

$$\begin{aligned} y_i^{k+1} &= \mathbf{prox}_{g_i/\rho}\left(L_i z^{k+1} + l_i + \lambda_i^k/\rho\right) \\ \lambda_i^{k+1} &= \lambda_i^k + \rho(L_i z^{k+1} + l_i - y_i^{k+1}) \ . \end{aligned}$$

Finally, Step (B.3) results to $w^{k+1} = \rho y^{k+1}$. Substituting w^{k+1} back to the augmented Lagrangian in (B.6), we end up with the ADMM iterations.

C

Stopping Conditions

We make use of the KKT conditions in order to terminate the algorithms presented in this work. The procedure is the following: We write down the optimality conditions for the Lagrangian (L) or for the saddle function (S) and express them in terms of the optimality conditions derived from each iterate of the corresponding algorithms. The algorithm is terminated once the KKT conditions are satisfied to some prespecified accuracy. More specifically we have:

AMA Consider the Lagrangian (L). The KKT conditions are:

$$0 = L_i z^\star + l_i - y_i^\star, \quad i = 1, \dots, M \tag{C.1}$$

$$0 = \nabla f(z^\star) + \sum_{i=1}^{M} L_i^T \lambda_i^\star \tag{C.2}$$

$$0 \in \partial g_i(y_i^\star) - \lambda_i^\star, \quad i = 1, \dots, M \tag{C.3}$$

Taking the optimality condition for Step 1 of Algorithm 1, we have that

$$\begin{aligned}
&\nabla f(z^{k+1}) + \sum_{i=1}^{M} L_i^T \lambda_i^k = 0 \\
&\nabla f(z^{k+1}) + \sum_{i=1}^{M} L_i^T \lambda_i^{k+1} + \sum_{i=1}^{M} L_i^T(\lambda_i^k - \lambda_i^{k+1}) = 0 \ ,
\end{aligned}$$

hence condition (C.2) is satisfied if $L^T(\lambda^{k+1} - \lambda^k) = 0$. Accordingly we have for Step 2 that

$$\begin{aligned}
&\partial g_i(y_i^{k+1}) + \lambda_i^k + \rho(L_i z^{k+1} + l_i - y_i^{k+1}) \ni 0 \\
&\partial g_i(y_i^{k+1}) + \lambda_i^{k+1} \ni 0 \ ,
\end{aligned}$$

which means that condition (C.3) is always satisfied each time Step 2 is executed. Finally, the primal optimality condition reads $L_i z^{k+1} + l_i - y_i^{k+1} = 0, \ i = 1, \dots, M$. We can thus write the primal and dual residuals as

$$r^{k+1} = Lz^{k+1} + t - y^{k+1} \tag{C.4}$$

$$s^{k+1} = L^T(\lambda^{k+1} - \lambda^k) \ . \tag{C.5}$$

PDA Consider the saddle function (S) with $h(z) = (1/2)z^T Qz + c^T z$. The KKT conditions are:

$$0 = Qz^\star + c + \partial\delta_{\mathcal{D}}(z^\star) + \sum_{i=1}^{M} L_i^T \lambda_i^\star \tag{C.6}$$

$$0 \in \partial g_i^\star(p_i^\star) - L_i z^\star - l_i, \quad i = 1, \dots, M \tag{C.7}$$

As before, we write down the optimality conditions for each step of Algorithm 2. The residuals read

$$r^{k+1} = (Q - (1/\tau^k)I)(z^{k+1} - z^k) + L^T(\lambda^{k+1} - \lambda^k) \tag{C.8}$$

$$s^{k+1} = P^k(\lambda^k - \lambda^{k+1}) + L(z^{k+1} - z^k) \tag{C.9}$$

where $P^k = \mathbf{diag}(\frac{1}{\rho_1^k}, \dots, \frac{1}{\rho_M^k})$.

ADMM Writing the optimality conditions for each step of Algorithm 3, we derive the formulas for the primal and dual residuals,

$$r^{k+1} = Lz^{k+1} + l - y^{k+1}, \tag{C.10}$$

$$s^{k+1} = \rho L^T(y^k - y^{k+1}) \ , \tag{C.11}$$

as given in [13].

Remark C.1. When the preconditioned versions of the algorithms are considered (Chapter 4), the residuals can be expressed in terms of the scaled variables in a similar manner.

References

[1] Hedy Attouch, Jérôme Bolte, Patrick R
Alternating proximal algorithms for weal
problems. Applications to dynamical gan
2008.

[2] Ahmet BehÃğet AÃğikmese and Lars
cation of a class of optimal control pro
constraints. *Automatica*, 2011.

[3] Heinz H Bauschke and Patrick L. Combe
tone operator theory in Hilbert spaces. Sp
2011.

[4] A. Beck and M. Teboulle. A fast iterat
rithm for linear inverse problems. *SIAM*

[5] Amir Beck and Marc Teboulle. Smooth
unified framework. *SIAM Journal on Op*

[6] Amir Beck and Marc Teboulle. A fast d
for convex minimization and application
2014.

[7] Stephen Becker and Jalal Fadili. A q
method. In *Advances in Neural Informat*

[8] Dimitri P. Bertsekas. *Constrained Optim*
Methods (Optimization and Neural Com
tific, 1996.

[9] Dimitri P Bertsekas. *Nonlinear Program*

[10] Dimitri P. Bertsekas and John N. Tsit
Computation: Numerical Methods. Pren

[11] Radu Ioan Bot, Ernö Robert Csetnek,
convergence rate improvement of a pri
solving monotone inclusion problems. *c*
2013.

[12] S.P. Boyd and L. Vandenberghe. *Conve*
versity Press, 2004.

[13] Stephen Boyd, Neal Parikh, Eric Chu, B
stein. Distributed optimization and stat
ing direction method of multipliers. *Fou*

[14] Stephen Boyd, Lin Xiao, Almir Mutapc
on decomposition methods. *Notes for*
2007.

[15] A.M. Bradley. Algorithms for the Equi
Application to Limited-Memory Quasi-
Stanford ICME, 2010.

[16] A. Chambolle and T. Pock. A first-orde
vex problems with applications to ima
Imaging and Vision, 2011.

[17] Gong Chen and Marc Teboulle. A proxim
for convex minimization problems. *Mat*

[18] Eric Chu, Brendan OâĂŹDonoghue, Ne
Primal-Dual Operator Splitting Method

[19] Patrick L Combettes and Jean-Christo
methods in signal processing. In *Fixed-p*
lems in science and engineering, pages 18

[20] Patrick L Combettes and Jean-Christoph
algorithm for solving inclusions with mi
and parallel-sum type monotone operat
analysis, 20(2):307–330, 2012.

[21] Patrick L Combettes and Băng C Vũ. V
splitting with applications to monotone
tion, 63(9):1289–1318, 2014.

[22] Patrick Louis Combettes, Laurent Cond
Vu. A forward-backward view of some pr
in image recovery. In *Image Processing (*
Conference on, pages 4141–4145. IEEE,

[23] Laurent Condat. A primal-dual splitting
involving Lipschitzian, proximable and l

[24] G. B. Dantzig and P. Wolfe. Decompo
grams. *Operations Research*, 1960.

[25] Wei Deng and Wotao Yin. On the gl
the generalized alternating direction met
technical report TR12-14, 2012.

[26] Olivier Devolder, François Glineur, and
ing technique for large-scale linearly co
SIAM Journal on Optimization, 22(2):7

[27] Alexander Domahidi, Aldo U. Zgraggen,
fred Morari, and Colin Neil Jones. Effi
multistage problems arising in receding

[28] J. Douglas and H. H. Rachford. On the
duction problems in two and three spac
1956.

[29] Jonathan Eckstein and Dimitri P. Berts
Splitting Method and the Proximal Poin
tone Operators. *Mathematical Programm*

[30] Ernie Esser, Xiaoqun Zhang, and Tony
for a class of first order primal-dual alg
in imaging science. *SIAM J. Img. Sci.*,

[31] J.E. Esser. *Primal Dual Algorithms for*
to Image Restoration, Registration and

[32] Hugh Everett. Generalized Lagrange Mul
lems of Optimum Allocation of Resourc

[33] H. J. Ferreau, H. G. Bock, and M. Dieh
to overcome the limitations of explicit
Robust and Nonlinear Control, 2008.

[34] Christopher Fougner and Stephen Boyd
conditioning for a graph form solver. *a*
2015.

[35] D. Gabay and B. Mercier. A dual algorit
variational problems via finite-element
Appl., 1976.

[36] Euhanna Ghadimi, André Teixeira, Iman
Optimal parameter selection for the alter
tipliers (ADMM): quadratic problems.
2013.

[37] P. Giselsson and S. Boyd. Monotonicity a
ods. In *Decision and Control (CDC), 201*
on, pages 5058–5063, Dec 2014.

[38] Pontus Giselsson and Stephen Boyd.
Rachford splitting and ADMM. *53rd IE*
Control, 2014.

[39] Pontus Giselsson and Stephen Boyd. Met
backward splittingâŃĘ. *Automatica*, 201

[40] R. Glowinski and A. Marrocco. A Moc
Method for Search of Saddle Points. 197

[41] R Glowinski and Patrick Le Tallec. *Augm*
splitting Methods In Nonlinear Mechan
Applied Mathematics, 1989.

[42] Tom Goldstein, Ernie Esser, and Richa
dual hybrid gradient methods for saddle
arXiv:1305.0546, 2013.

[43] Tom Goldstein, Brendan O'Donoghue, S
niuk. Fast alternating direction optimi
on Imaging Sciences, 7(3):1588–1623, 20

[44] Tom Goldstein and Stanley Osher. Th
regularized problems. *SIAM J. Imaging*

[45] Tom Goldstein, Christoph Studer, and Ri
forward-backward splitting with a fasta
arXiv:1411.3406, 2015.

[46] Osman Güler. On the convergence of th
convex minimization. *SIAM J. Control*
1991.

[47] O. GuÌĹler. New proximal point algorit

[48] E.N. Hartley, J.L. Jerez, A. Suardi, Jan
and G. Constantinides. Predictive Con
plication to Aircraft Control. *IEEE T*
Technology, 2013.

[49] Bingsheng He and Xiaoming Yuan. Conv
Algorithms for a Saddle-Point Problem:
SIAM J. Imaging Sciences, 2012.

[50] Bingsheng He and Xiaoming Yuan. C
dual algorithms for a saddle-point proble
SIAM Journal on Imaging Sciences, 5(1

[51] Bingsheng He and Xiaoming Yuan. On t
the Douglas-Rachford Alternating Direct
Analysis, 2012.

[52] B.S. He, H. Yang, and S.L. Wang. Alt
self-adaptive penalty parameters for mo
Journal of Optimization Theory and Ap

[53] R.M. Hestenes. Multiplier and gradient
tion Theory and Applications, 1969.

[54] L. Hurwicz K. J. Arrow and H. Uzawa.
programming. *Stanford University Press*

[55] T. Chan M. Zhu. An efficient primal-du
total variation image restoration. UCLA

[56] Jacob Mattingley and Stephen Boyd. C
embedded convex optimization. *Optimiz*

[57] Zhi-Quan Luo Mingyi Hong. On the linea
direction method of multipliers, 2012.

[58] Ion Necoara and Andrei Patrascu. Du
dual first order algorithm for quadratic
arXiv:1504.05708, 2015.

[59] Ion Necoara and Johan AK Suykens. A
nique to decomposition in convex optimiz
Transactions on, 53(11):2674–2679, 2008

[60] ArkadiÄŋ NemirovskiÄŋ and David Bo
complexity and method efficiency in op
series in discrete mathematics. Wiley, C
Wiley-Interscience publication.

[61] Yu. Nesterov. A method for solving th
with convergence rate $o(1/k^2)$. *Dokl. Ak*

[62] Yu. Nesterov. *Introductory lectures o*
course. 2004.

[63] Yu. Nesterov. Smooth minimization o
Program., 2005.

[64] Yurii Nesterov. How to advance in struc
TIMA: Mathematical Programming Soci

[65] Yurii Nesterov et al. Gradient methods
tive function, 2007.

[66] Robert Nishihara, Laurent Lessard, Ben
and Michael I. Jordan. A general analys
In *International Conference on Machine*

[67] Jorge Nocedal and Stephen J Wright. *N*
tion). Springer, 2006.

[68] B. O'Donoghue and E.J. Candes. Adapt
dient Schemes. *arXiv.org*, 2012.

[69] B. O'Donoghue, G. Stathopoulos, and S
optimal control. *IEEE Transactions o*
2012.

[70] Brendan O'Donoghue, Eric Chu, Neal Pa
tor splitting for conic optimization via ho
arXiv preprint arXiv:1312.3039, 2013.

[71] N. Parikh. Proximal operators. https://

[72] Neal Parikh and Stephen Boyd. Proxim
Trends in Optimization, 2014.

[73] Panagiotis Patrinos and Alberto Bem
gradient-projection algorithm for embed
trol. *Automatic Control, IEEE Transact*

[74] Thomas Pock and Antonin Chambolle
first order primal-dual algorithms in co
Vision (ICCV), 2011 IEEE Internation
1769. IEEE, 2011.

[75] B.T. Polyak. Some methods of speeding
methods. *USSR Computational Mathem*
, 4(5):1 – 17, 1964.

[76] L. Popov. A Modification of the Arrow
Saddle Points, journal = Mathematical

[77] M. J. D. Powell. A method for nonlin
problems. In R. Fletcher, editor, *Optimi*
Press, 1969.

[78] A. U. Raghunathan and S. Di Cairano. In
ing direction method of multipliers for
Decision and Control (CDC), 2014 IEI
pages 5819–5824. IEEE, 2014.

[79] A.U. Raghunathan and S. Di Cairano. C
ternating direction method of multipliers
and model predictive control. In *Inter*
matical Theory of Networks and System
2014.

[80] R. T. Rockafellar. Augmented Lagrangia
imal Point Algorithm in Convex Progra
tions Research, 1976.

[81] R. Tyrrell Rockafellar. Monotone Ope
Algorithm. *SIAM J. on Control and Op*

[82] Ron Shefi and Marc Teboulle. Rate of Co
sition Methods Based on the Proximal M
Minimization. *SIAM Journal on Optim*

[83] N. Z. Shor, Krzysztof C. Kiwiel, and And
Methods for Non-differentiable Functio
Inc., 1985.

[84] R. Sinkhorn and P. Knopp. Concerning n
stochastic matrices. *Pacific J. Math.*, 21

[85] P. Tseng. Applications of splitting algo
vex programming and variational inequa
1991.

[86] P. Tseng. On accelerated proximal gradi
optimization. *submitted to SIAM Journ*

[87] E. Ullmann. A Matlab toolbox for C-
methods. Master's thesis, ETH Zurich,

[88] Băng Công Vũ. A splitting algorithm f
volving cocoercive operators. *Adv. Com*

[89] L. Vandenberghe. Optimization methods
EE 236C lecture notes, 2010.

[90] Xiaojie Xu, Pi fang Hung, and Yinyu Ye
self-dual linear programming algorithm
of Operations Research, 1996.

[91] Yinyu Ye, Michael J Todd, and Shinji
homogeneous and self-dual linear progra
of Operations Research, 1994.

[92] M. Zhu and T. Chan. An efficient primal
for total variation image restoration. *U(*